Fast Speaker Independent Large Vocabulary Continuous Speech Recognition

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
der Fakultät für Informatik
an der Universität Karlsruhe (Technische Hochschule)
vorgelegte

Dissertation

von
Monika Woszczyna
aus Brüssel

Day of the oral exam:	February 13th, 1998
Advisor:	Prof. Dr. A. Waibel
Co-Advisor:	Prof. Dr. W. Ward

Abstract

To build useful applications based on large vocabulary continuous speech recognition systems, such systems have to run in real time on common platforms. However, with most research focused on further reducing the recognition error rates, the topic of speed has been neglected in the development of speech recognition algorithms.

I will present a speaker independent system that has been designed for fast speech recognition using vocabularies up to 65,000 words. Using the approaches presented in this thesis, this recognizer can now run in real time, 200 times faster than the original evaluation system.

Important progress was made on the following topics:

Tradeoffs: a better understanding of the tradeoffs between the computational effort and the accuracy of the acoustic modeling provides a foundation to methodically develop faster algorithms.

Algorithms: a number of new or improved algorithms were introduced and analyzed in this work, such as:

- *Lookaheads:* Lookaheads provide early estimates of acoustic and language model scores to the recognition process. These estimates can be used to restrict the search space at low risk.
- *Generalized-BBI:* the cross codebook *bucket box intersection* algorithm reduces the effort for the observation probability computation, especially when used in speech recognition systems with very large continuous *hidden Markov models*.
- *Frame Skipping:* for many sections of the recorded speech, not all operations have to be performed for all frames provided by the preprocessing.

The recognizer presented here is derived from existing evaluation systems. No specialized small recognizer had to be trained in order to get real time performance. Therefore, the new algorithms can also help to reduce the cycle time in developing new evaluation systems.

All algorithms have been developed and tested on a state-of-the-art dictation system for the *North-American-Business-News (NAB)* Task. By testing on other databases, robustness with respect to new microphones and foreign accents has been verified. It was also shown that the new algorithms can be ported to a variety of tasks such as a Japanese dictation system and the automatic transcription of spontaneously spoken German scheduling dialogues.

Zusammenfassung

Um sinnvolle Anwendungen auf der Basis von sprecherunabhängiger Spracherkennung mit großen Vokabularen bereitstellen zu können, müssen diese Spracherkennungssyteme in Echtzeit lauffähig sein. Weil das Augenmerk der Spracherkennungs-Forschung hauptsächlich auf der Senkung der Fehlerrate lag, wurde die Entwicklung schneller Algorithmen oft vernachlässigt.

Ich stelle ein System vor, das für schnelle Spracherkennung bei Vokabularen von bis zu 65,000 Wortformen entwickelt wurde. Durch die in dieser Arbeit vorgestellten Ansätze läuft dieser Erkenner inzwischen in Echtzeit, 200 mal schneller als das ursprüngliche Evaluationssystem.

Wichtige Fortschritte wurden auf den folgenden Gebieten gemacht:

Wechselwirkungen: ein besseres Verständnis der Wechselwirkungen zwischen dem Aufwand und der Genauigkeit für die statistische Modellierung des Signals erlaubt die systematische Entwicklung schnellerer Algorithmen.

Algorithmen: mehrere neue oder verbesserte Algorithmen wurden im Rahmen dieser Arbeit eingeführt und analysiert, zum Beispiel:

- *Lookaheads:* Lookaheads geben frühe Schätzungen für die Bewertungen der akustischen Modelle und der Sprachmodelle. Diese Schätzungen können verwendet werden, um den Suchraum mit minimalem Risiko zu beschränken.
- *Generalized-BBI:* der codebuchübergreifende *Bucket-Box-Intersection*-Algorithmus reduziert den Aufwand zur Berechnung der Emissionswahrscheinlichkeiten, insbesondere bei der Verwendung von großen, voll kontinuierlichen *Hidden-Markov-Modellen.*
- *Frame-Skipping:* Für viele Teile des Sprachsignals, ist es nicht notwendig alle Berechnungen für jedes einzelnde Zeitfenster (Frame) zu berechnen.

Der hier vorgestellte Erkenner wurde von einem bestehenden Evaluationssystem abgeleitet. Es wurde bewußt darauf verzichtet, eigens einen speziellen, kleineren Erkenner zu trainieren um Echtzeitverhalten zu erzielen. Deshalb können die neuen Algorithmen auch dazu verwendet werden die Zykluszeit bei der Entwicklung von Evaluationssystemen zu senken.

Alle Algorithmen wurden auf einem *'state-of-the-art'* Diktiersystem für das *North-American-Bussiness-News* Szenario entwickelt und getestet. Durch Vergleichstests auf anderen Datenbasen wurde die Robustheit gegenüber anderen Mikrophonen und Sprechern mit Akzent getestet. Die Portierbarkeit der Algorithmen wurde anhand der Anwendung auf so verschieden Szenarios wie einem Japanischen Diktiersystem und einem System zur automatischen Transliteration spontan gesprochener Terminabsprache-Dialoge überprüft.

Contents

1 Introduction **9**

1.1 Why Speech Recognition 9

1.2 Recognition Speed versus Accuracy 10

1.3 How to Read this Thesis 12

2 Fundamentals of Speech Recognition **13**

2.1 Measuring Performance 14

2.1.1 Recognition Accuracy 15

2.1.2 Recognition Speed 16

2.2 Modeling the Speech Signal 17

2.2.1 Language Models 17

2.2.2 Sampling and Feature Vectors 18

2.2.3 Phonemes and Pronunciation Dictionaries 21

2.2.4 Hidden Markov Models 22

2.2.5 Observation Probabilities 23

2.2.6 Forward Algorithm 23

2.2.7 Viterbi Algorithm 25

2.3 Continuous Speech Recognition 26

2.3.1 Time Synchronous Search 27

2.3.2 Depth First Search 30

3 Test Environment **31**

3.1 The JANUS Recognition Toolkit 31

3.2 The *North American Business News* Task 33

3.3 The Baseline NAB Recognizer 34

4 Speed Issues in Speech Recognition **35**

4.1 Language Models 36
4.1.1 Smoothed Language Models 36
4.1.2 Delayed Bigrams 37
4.2 Search 39
4.2.1 Pruning the Search Space 39
4.2.2 Tree Search 41
4.2.3 Forward-Backward Search 43
4.2.4 Word Graphs 44
4.3 Observation Probabilities 45
4.3.1 Nearest Neighbor Approximation 45
4.3.2 Gaussian Selection and related approaches 46
4.3.3 Neural Nets 48
4.4 Parallel Computing and Hardware 49
4.4.1 Threads 49
4.4.2 Special Purpose Hardware 50
4.4.3 Faster Processors 51

5 Thesis Contributions **55**

5.1 Search 56
5.1.1 Cross-Word Models 56
5.1.2 Tree Search 58
5.1.3 Linear Search 59
5.1.4 Pruning the Search Space 61
5.1.5 Unigram Lookaheads 69
5.1.6 Phoneme Lookaheads 71
5.2 Observation Probabilities 76
5.2.1 Order and Importance of Gaussians 77
5.2.2 Reduced Dimensionality 78
5.2.3 Radial Covariances 81
5.2.4 Low Level Optimizations 81
5.2.5 Bucket Box Intersection 82
5.2.6 Generalized-BBI 87

5.3 Skipping Input Frames 96
5.3.1 Skipping Score Computations 96
5.3.2 Reducing the Frame Rate 98
5.3.3 Conditional Frame Skipping 99

6 Application Oriented Issues **101**
6.1 Portability 101
6.2 Robustness 105
6.3 Pipelining 107
6.4 The NAB Dictation System 108

7 Conclusion **111**
7.1 Summary 111
7.2 Assessment 112
7.3 Perspectives 113

A Other LVCSR Systems **115**
A.1 AT&T 115
A.2 BBN – Byblos 116
A.3 Cambridge University – ABBOT 118
A.4 Cambridge University – HTK 119
A.5 Carnegie Mellon University: SPHINX 120
A.6 Daimler Benz 121
A.7 Dragon 121
A.8 IBM 122
A.9 LIMSI 123
A.10 Microsoft – Whisper 124
A.11 MIT – Lincoln Laboratories 125
A.12 Philips and University of Aachen 125
A.13 Siemens 126
A.14 SRI – DECIPHER 126
A.15 Other LVCSR Systems 127

Bibliography **129**

Chapter 1

Introduction

1.1 Why Speech Recognition

For most common tasks, speech is a much more natural way to communicate than typing on a keyboard. Most people learn to speak long before they learn to write or type. According to official statistics [World Factbook, 1996] more than three percent of all Americans and Germans cannot read or write at all.

While there are typists that can type faster than they can speak, this is certainly not true for most of us. Employing a typists is often considered to be too expensive. However, the time of the people who might use a typist is usually even more valuable. Since word processing is the most common use of PCs, the potential market for fast dictation systems is large. Many companies such as Microsoft [Alleva et al., 1997], IBM [Gopalakrishnan et al., 1995], Dragon [Ellermann et al., 1993], and Siemens [Niemöller et al., 1997] put substantial resources into the development of such products.

Other than being more economical, computers can also be programmed to be more discreet than humans. Applications that require this sort of discretion could be the dictation of business letters, speech-to-speech translation [Osterholtz et al., 1992, Bub et al., 1997], and the monitoring of telephone lines.

Just like the miles of tapes from tapped phone lines all over the world, large amounts of data are not accessible as texts. Less controversial examples are broadcast news, interviews, and talk shows. To exploit these sources, a summary or a key word index is required to allow the search of speech documents the same way as collections of filed text documents. Using a speech recognition system, such databases can be created and maintained automatically [Hauptmann and Wactlar, 1997].

While typing, the user has to stay put at the computer, have both hands on the keyboard and the eyes focused on the screen. These limitations are known to cause health problems such as shortsightedness, headaches, back pain and repetitive strain injury [RSI, 1997]. Also, they restrict the use of a computer to applications where

the user does not need to move around, and does not needs hands and eyes for other purposes. Speech offers an ergonomic input alternative and it allows drivers to keep both hands on the wheel while accessing the navigation database of their car.

Finally, the space requirements of a microphone are substantially lower than those of a full size keyboard. Considering that keyboards tend to be among the largest parts of a modern computer this could also become an issue when designing the next generation of palm-tops.

Limitations

Speech recognition is not always the best way to interact with a computer, just as using a computer is often not the best way to solve a problem. Continuous speech recognition is still too expensive to be integrated into a cheap pocket calculator. It is also impractical to have fifty people talking to their computers in an open-plan office. Entering the PIN for a MAC-card over a speech interface raises obvious security problems. Adverse environments such as the noise level in a cockpit limit the usability of speech recognition interfaces in some situations. Finally, some drawing and design tasks cannot be efficiently solved using speech recognition as the only input modality.

However, combinations with related techniques provide interesting ways to extend the field of useful speech applications [Waibel et al., 1997]. Speaker identification can be used to add security, lip reading raises the accuracy in environments with high noise levels, and touch-screens with gesture recognition help to design complex objects.

1.2 Recognition Speed versus Accuracy

Problem

Many of the applications mentioned above require real time large vocabulary speech recognition. This means the recognition process should not take longer on a common computer than the speaker took to utter the sentence.

Unfortunately, the complexity of speech recognition systems has been steadily increasing along with the recognition accuracy — more often than not at the expense of recognition speed. The reason for this development is associated with the sole evaluation criterion for speech recognition systems: the word accuracy on benchmark tasks. With this paradigm, the recognition speed is only relevant where it limits the number of experiments that can be performed to raise the word accuracy. To achieve peak performance, common recognizers used in such evaluations take several hundred seconds to process a single second of input speech on state-of-the-art computers.

However, for most real-world applications, recognition speed is just as important as recognition accuracy. A dictation system that takes four days to process half an hour of speech will not make a useful product. A database interface that takes 15 minutes to respond to a four second query is not likely to be accepted by any user.

The argument that speed is really no issue as programs get faster with each new generation of computers is only partly valid. On one hand, systems tend to get more complex at a higher rate than computers get faster. On the other hand, a speech recognition system used as an input interface is not supposed to devour 100% of the available cycles.

The importance of speed and efficient algorithms in speech recognition can best be illustrated by the substantial research effort of companies such as Microsoft [Alleva et al., 1997], IBM [Gopalakrishnan et al., 1995], Dragon [Ellermann et al., 1993], and Siemens [Niemöller et al., 1997] on this topic.

Solution Concept

One goal of this thesis work was to investigate how the recognition speed of HMM-based LVCSR systems can be improved. To that end, the recognition time of a given prototype recognizer was to be cut by a factor of 200 to real time performance without reducing the accuracy below a set threshold. All progress was monitored by evaluating the performance of the resulting system using recordings that were collected for common recognition scenarios.

In many cases the faster algorithms and techniques include approximations that reduce the recognition accuracy of the system. The resulting speed-up and accuracy loss often depend on a number of parameters. The behavior of a system that combines several lossy techniques can be complex and is studied in more detail in the following chapters. Understanding how much and why the accuracy is influenced by introducing new algorithms into the recognition process gives valuable clues for further research.

To guide the research, the distribution of the recognition time over the subtasks of the recognition process was analyzed. All subtasks that were significantly contributing to the computational effort were studied in detail. After exploiting the most promising improvements found in the literature, new algorithms were developed and introduced to further reduce the time spent on the corresponding subtasks.

Using the spin-offs of the research presented in this thesis, a dictation system with a vocabulary of 65,000 words was built. This dictation system runs in real time, 200 times faster than the original full scale evaluation recognizer.

1.3 How to Read this Thesis

This document has been structured to allow direct access to relevant information while still being readable from front to back. To avoid unwanted repetitions, the author has used the margins to add pointers with page numbers to related chapters (these are links in the HTML version of this thesis).

→ 9 **1 Introduction:** The first half of this introduction motivates speech recognition research and highlights the importance of fast speech recognition.

→ 13 **2 Fundamentals of Speech Recognition:** This chapter is an introduction to the fundamentals of speech recognition, insofar as they are relevant for later chapters. Speech experts may want to skip this chapter.

→ 31 **3 Test Environment:** Here the environment used to develop and test most algorithms is described: the *North-American-Business-News* scenario and database, the JRTk speech recognition toolkit, and the baseline recognizer.

→ 35 **4 Speed Issues in Speech Recognition:** This chapter contains a brief overview of some common techniques that have been used in the past to speed up recognition systems. For some of the techniques that have been applied to the JRTk system performace results are presented.

→ 55 **5 Thesis Contributions:** This chapter describes the algorithms and techniques used and developed for the work presented in this thesis, such as the search-engine, the *unigram lookaheads*, the HMM-based *phoneme lookaheads*, and the *Generalized-BBI* algorithm. Experimental results as well as some implementation specific details of the recognizer used for the experiments are added where appropriate.

→ 101 **6 Application-Oriented Issues:** Other than the speed issues addressed in the last two chapters, there are a number of further requirements to make a speech recognition system usable. Robustness and portability are two of the most important. The third section describes how the recognizer can be run as a pipeline to produce output while the speaker is speaking. In the last section, the author presents a dictation system based on the research work presented in earlier chapters.

→ 111 **7 Conclusion:** Finally, on page 111, there is a summary of the results to put the achievements of this thesis into the context of the findings of other groups. A brief outlook on the future of fast speech recognition closes this chapter.

→ 115 **Appendix: Other LVCSR Systems:** This appendix provides an overview of large vocabulary continuous speech recogniton systems other than the one used for this thesis. Whereever such a system is mentioned in the above chapters, this appendix can be used as quick reference to find additional information.

Chapter 2

Fundamentals of Speech Recognition

When the weight of the paperwork
equals the weight of the plane,
the plane will fly.
– Donald Douglas

This chapter provides an introduction to the fundamentals of speech recognition, insofar as they are relevant for later chapters.

The topics covered in this chapter are: training data, test data, evaluation data, word correct rate, word error rate, word accuracy, real-time, CPU-time, language models, bigrams, trigrams, sampling, feature vectors, spectrum, pronunciation dictionaries, hidden Markov models, forward algorithm, Viterbi algorithm, continuous speech recognition, time synchronous search, depth first search, and N-best searches.

Techniques that have been introduced to speed up the recognition process, such as pruning strategies and using a tree organized lexicon in the search for the best word sequence, can be found in the chapter on speed issues. → 35

Speech recognition experts may want to skip this chapter and proceed directly to the description of the test and development environment. For further reading, → 31 [Waibel and Lee, 1990] and [Schukat-Talamazzini et al., 1993] provide good starting points.

This chapter is divided into three sections: First, the techniques used to measure recognition quality and recognition speed are introduced. The second section states the speech recognition problem, and explains what methods such as *statistical language modeling* and *hidden Markov models* can be used to solve it. Finally, using these methods, two approaches to continuous speech recognition are presented.

2.1 Measuring Performance

In computer science, it is a common approach to solve problems in several steps:

1. State and analyze the problem.
2. Set up a theory on how to solve the problem.
3. Derive experiments that can be used to support or disprove the theory.
4. Evaluate the experimental evidence.

In speech recognition, the crucial question that needs to be answered by experiments is '*will the suggested approach improve the recognition of new data ?*'.

It is important that the data used to verify a theory has not been used in any form while building the recognizer. Therefore, there are usually three sets of data used to build a recognition system: training data, test data, and evaluation data.

The *training data* is used to build the recognizer and adjust its many parameters. The amount of required training data depends on the recognition approach. For most recognition systems, more training data will yield better recognition results.

The *test data* is used to evaluate new algorithms during the development phase of the recognizer. Since many decisions are made based on this data (e.g. whether to use algorithm A or B in the system), it becomes '*contaminated*'; the decisions may not be independent of the test set, and the resulting performance may be higher on the test data than on completely unseen data.

The *evaluation data* used for the final assessment of the system should be therefore be unseen. This means none of the system's parameters has been adjusted to this data, and no decision was ever made based on recognition results on this data. The amount of test and evaluation data influences the confidence in the results, that is the expected maximum deviation from the original results when testing on a different test set.

Throughout this thesis, the problem can usually be formulated as '*the recognizer is too slow*'.
While there are many suggested solutions to this problem, the experiments used to validate these theories are mostly based on the recognition accuracy and the recognition time measured on actual speech data as evaluation criteria.

2.1.1 Recognition Accuracy

The *Word Correct Rate* for continuous speech recognition is given as:

$$\textit{Word Correct Rate} = 100 \cdot \frac{\text{number of correctly recognized words}}{\text{number of spoken words}} \tag{2.1}$$

Words that are inserted by mistake are not included in computing the *Word Correct Rate*. The more insertion errors are made in one sentence, the more likely one of those matches a word that was actually spoken, though the usability of the recognition output is low. This is why the *Word Error Rate* and the *Word Accuracy* are more commonly used to measure the recognition quality.

$$\textit{Word Error Rate} = 100 \cdot \frac{\text{substitutions} + \text{deletions} + \text{insertions}}{\text{number of spoken words}} \tag{2.2}$$

The *Word Accuracy*[1] is defined as $100 - \textit{Word Error Rate}$.

Confidence Intervals

The *Word Correct Rate* is a mean over a closed set of two values: each recognized word can be either correct (1) with a probability p or wrong (0) with a probability $(1 - p)$.

According to [Bamberg and Baur, 1996], the boundaries $V_{o/u}$ of the confidence interval for such a distribution are computed to:

$$V_{o/u} = \frac{2n\bar{X} + c^2 \pm c\sqrt{4n\bar{X}(1-\bar{X}) + c^2}}{2(n + c^2)} \tag{2.3}$$

Here, n is the number of words in the test set and $\bar{X}$ the mean of the *Word Correct Rate*. Also, $c = 1.96$ for a 95% confidence interval given $n > 100$ can be found by table lookup or numerical integration.

To decide whether the output of two recognition runs on the same test data differ significantly, the distribution of errors within the test set can be analyzed [Gilick and Coz, 1989]. For each recording in the test set, the difference in *Word Accuracy* between both tests is calculated. It is then assumed that the deviation from 0 of the mean over these values is caused by random fluctuations. Unless this hypothesis can be disproved, the difference between the outputs is not significant.

[1] An important extension of quality assessment for recognition of spontaneous speech is given in [Fisher and Fiscus, 1993]; to avoid random matches where the recognition is completely wrong, this measure takes into account not only what words were recognized but also where in the recording they were recognized. For low Error Rates in dictation systems this extension does not make a significant difference.

2.1.2 Recognition Speed

The recognition speed is defined as the required recognition time in seconds per second of speech input. The recognition time can be measured in *'real'* seconds or in CPU-seconds, which is the time the recognition process has exclusive use of the central processing unit of a computer with a multitasking operating system such as UNIX. Since the time used for network access or loading virtual memory pages from disk is not counted in the CPU-seconds, care must be taken that new algorithms do not increase the real recognition time while reducing the CPU-time. The advantage of using CPU-seconds is that they are mostly independent of the amount of core memory available and the load other processes put on the CPU, making comparisons between separate test runs easier.

Quite obviously, the recognition time depends on the computer and compiler used
→ 51 for the experiment. More information on this topic can be found in chapter 4.4.3.

In the following chapters, the recognition time is always given in CPU-seconds. Unless the recognition process is explicitly divided into several steps, the recognition time always covers all parts of the recognition process, including all file I/O and preprocessing.

As a unit for the recognition time I will use the *real time factor* (RT). 2 RT means that the computer needed twice as long to recognize a recording than the speaker took to say it. Ideally, the time during which the speech is recorded is used for
→ 107 speech recognition. If a realtime system is pipelined, the time delay between the end of the input speech and the recognition can become neglegible.

Confidence Intervals

To compute the confidence intervals for the recognition time, the test set is subdivided into a sufficiently large number of segments. If all errors in one segment are assumed to be independent from the errors in the remaining segments, the variance σ and the mean $\bar{X}$ of the recognition speed can be computed assuming a normal distribution.

The borders $V_{o/u}$ of the confidence interval are computed to

$$V_{o/u} = \bar{X} \pm \frac{\sigma c}{\sqrt{n}} \tag{2.4}$$

Where n is the number of recordings and $c = 1.96$ for $n > 100$ given a 95% confidence interval.

2.2 Modeling the Speech Signal

The goal of speech recognition is to find the most likely word sequence W for a given speech signal A. This is the word sequence for which

$$P(W|A) = \frac{P(W)P(A|W)}{P(A)} \tag{2.5}$$

is largest.

P(W) gives the probability of a particular word sequence, and is independent of the actual input signal. For instance, the word sequence *'but does it'* is much more likely to occur in any sentence than the sequence *'but doses it'*.

P(A|W) is the probability of observing the signal A given that the actual word sequence is W.

P(A) is the probability of the recorded signal. Once the signal has been recorded, $P(A)$ is the same for all word sequences that may have been said. Therefore, $P(W|A)$ is largest for the word sequence for which the product $P(W)P(A|W)$ is largest.

The following sections present methods to compute $P(W)$ and $P(A|W)$:

Language Models: shows a simple algorithm to estimate $P(W)$. → 17

Sampling and Feature Vectors: explains how the speech signal A is made accessible to the computer. → 18

Phonemes and Pronunciation Dictionaries: describes how the words in W can be broken into smaller characteristic units of speech (*phonemes*) to simplify the modeling of $P(A|W)$, the word. → 21

Hidden Markov Models: introduces the concept of representing phonemes as a chain of states s_j that emit feature vectors x_i as a generative model for $P(A|W)$. → 22

Observation Probabilities: gives a method to estimate the probability density $f(x_j|s_i)$ of a single feature vector being produced by a single state. → 23

Forward Algorithm: presents an algorithm to actually compute $P(A|W)$ for a given word sequence W using the methods introduced in the previous sections. → 23

Viterbi Algorithm: gives a faster algorithm that can be used to approximate $P(A|W)$ along the best state sequence through a given word sequence W. → 25

2.2.1 Language Models

To estimate the probability $P(W)$ of a given word sequence W, most recognition systems use n-grams such as bigrams and trigrams [Jelinek, 1990]. On a large text corpus the counts for all word pairs (for bigrams), triples (for trigrams) or sequences of words (for n-grams) are accumulated to compute the conditional probability

$P(w_i|w_{i-1}, w_{i-2}, .., w_{i-n+1})$ for observing a word w_i given its predecessors. Using trigrams, the probability $P(W)$ is computed as:

$$P_{\text{trigram}}(W) = \prod_{i=1}^{N} P(w_i|w_{i-1}, w_{i-2}). \tag{2.6}$$

To estimate probabilities for word pairs, triples or sequences that do not occur frequently enough in the chosen text corpus, a variety of smoothing and back-off techniques can be used [Katz, 1987, Ney and Essen, 1991].

Though many other approaches to language modeling have been proposed, few have been as successful as n-gram models for tasks where a large enough corpus for the probability estimation is available.

2.2.2 Sampling and Feature Vectors

Before the speech signal can be analyzed by the computer, the variations in pressure at the microphone have first to be transformed into an electrical signal. The resulting continuous signal is then cut into discrete time slices and transformed into discrete amplitude values by an analog to digital converter hardware. Common resolutions for the applications in this thesis are 16,000 16-bit values per second. For reference: Compact Disk players use two channels, with 48,000 16-bit values per second and channel.[2]

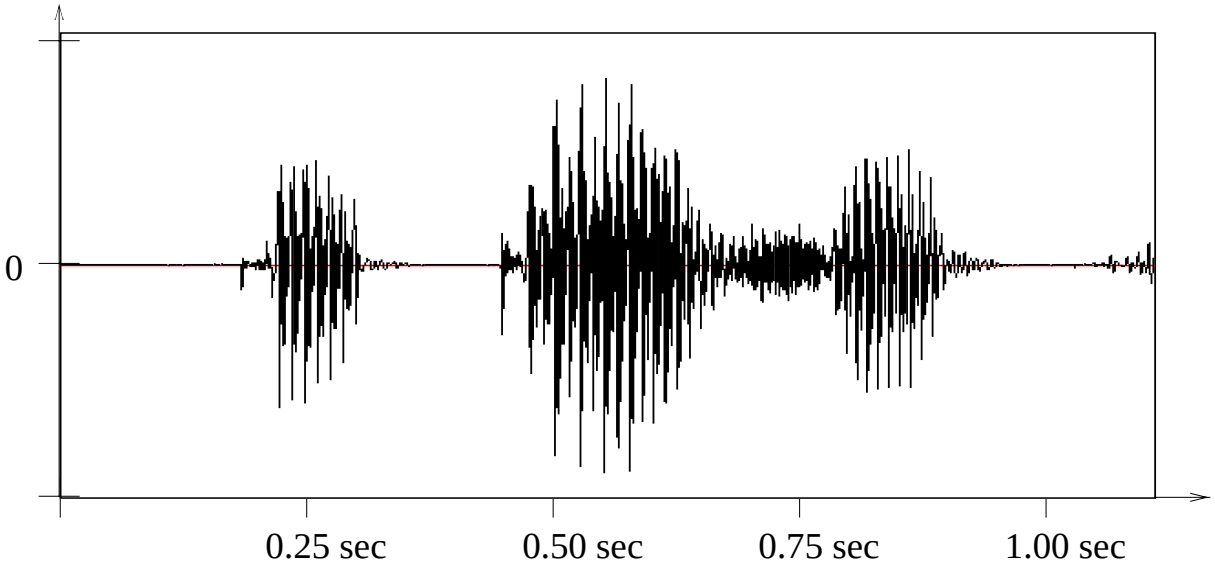


Figure 2.1: A picture of the speech signal '*but does it*' without preprocessing. The relative amplitude of the signal is given over the recording time in seconds.

[2] The highest frequency that can be reproduced from the sampled signal is a little smaller than half the sampling rate. While the region above 8 kHz contains little valuable information for speech recognition, hi-fi definitions require an accurate reproduction of frequencies up to 20 kHz.

It is not a good idea to try to use the sampled signal directly for speech recognition purposes. The amount of data is still too large (32 kBytes per second) and contains random noise as well as unwanted information on speaker properties, background noise and the recording channel. This interference complicates the recognition process.

A number of preprocessing steps are commonly used to reduce the amount of data and to emphasize the dependence on the spoken words by suppressing irrelevant information. While there are many algorithms that are frequently used for the preprocessing of the speech signal, this section only gives a brief introduction into the techniques that are required for a complete understanding of the work presented in this thesis.

Most preprocessing algorithms begin by computing a spectral analysis over a window covering N samples, which is moved over the signal in increments of 10 or 20 milliseconds. The time slices corresponding to these increments are called *frames*. The spectral coefficients $F_\mu^{(m)}$ for the frame starting at the mth sample of the original signal f are computed using the discrete *Fourier Transform*:

$$F_\nu^{(m)} = \sum_{n=0}^{N-1} f_{m-n} w_n e^{-2\pi i \nu n / N} \tag{2.7}$$

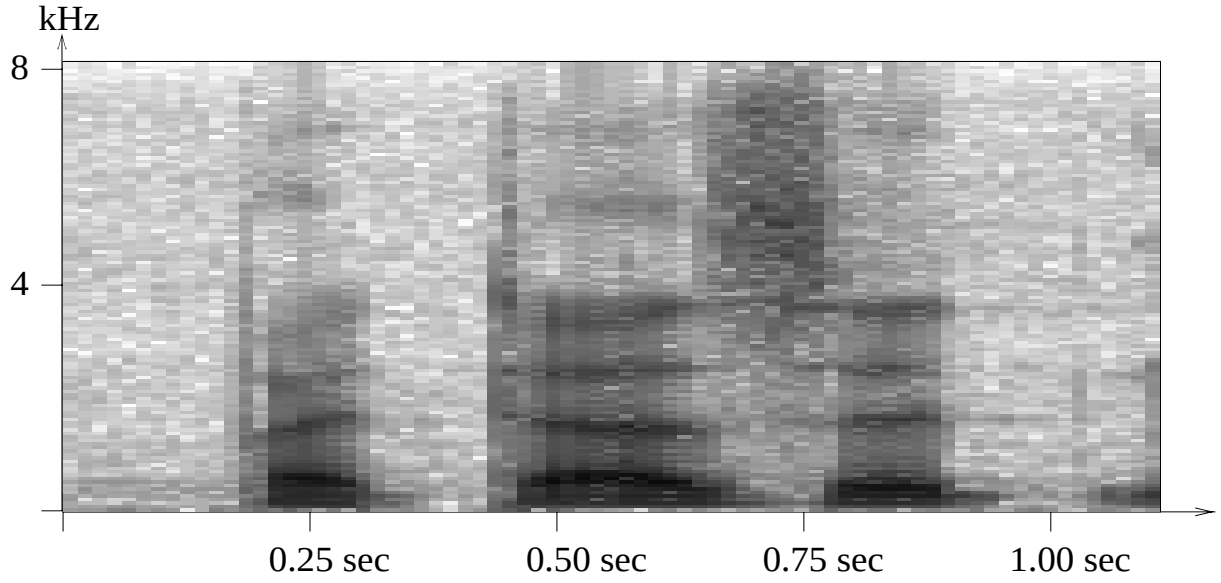


Figure 2.2: The spectrum of the signal '*but does it*'. Darker regions represent frequencies that contribute significantly to the signal at a given time. Unlike in the original signal, the phonemes `B` and `D` (marked by a sudden rise over all frequencies), the vowels (horizontal stripes representing overtones produced by the vocal cords), and the `S` (high frequencies only) can be easily spotted in the spectrum.

From the resulting spectral coefficients, a variety of features such as *Mel-scale* and *Mel-cepstrum* coefficients can be derived [Waibel and Lee, 1990]. Many of these

features are motivated by the spectral resolution and sensitiveness of the human ear.

Also, to capture the changes in the signal, a feature-vector used to describe the signal for one frame can additionally contain features of neighbor frames or the difference between features of the predecessor and successor frames.

The output of the preprocessing is usually a feature vector with 8 to 50 coefficients for every 6 to 10 milliseconds of speech. From now on, the signal A in equation 2.5 will be represented by such a sequence of feature vectors.

2.2.3 Phonemes and Pronunciation Dictionaries

To simplify the computation of $P(A|W)$, the words of the word sequence W can be broken into smaller characteristic units of speech, called *phonemes*.

Since many languages do not have simple rules to derive pronunciation from spelling or vice versa, this information is provided to the speech recognizer in the form of pronunciation dictionaries.

Instead of *phonemes*, often simplified phone-like units are used, which will be represented as symbols in capital letters (`AA, AY, B, ..`). For simplicity, they will still be referred to as phonemes.

```
...
aisles          AY L Z
eye             AY
dollar          D AA L AXR
quit            K W IH T
quite           K W AY T
to              T UW
too             T UW
two             T UW
...
```

```
...
Monat           M OH N A T
Pfunden         P F U N D E2 N

teilen          T AI L E2 N
teilgenommen    T AI L G E2 N O M E2 N
teilnehmen      T AI L N EH M E2 N
...
```

If there is more than one common pronunciation for a word, alternative variants can be introduced into the dictionary.

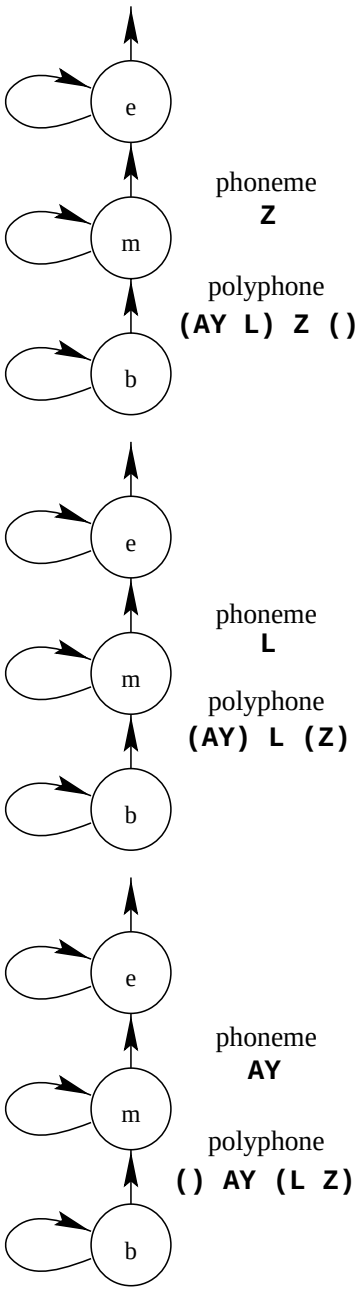
```
...
the             DH AX
the(2)          DH IY
...
```

```
...
wichtig         V I CH T I CH
wichtig(2)      V I CH T I K
...
```

2.2.4 Hidden Markov Models

To describe the speech process by means of statistics, the speech signal A is considered to be produced by a *Markov chain* with states and transitions: Each state can produce an observed feature vector with a certain probability. Using these probabilities, the probability $P(A|W)$ of a signal A being produced by the *Markov chain* corresponding to the word sequence W can be computed.

Since the speech signal differs between the borders and the center of a phoneme, each phoneme is commonly modeled as a sequence of three states representing the beginning, the center, and the end segment of the phoneme. Sometimes each of these segments has two states to control the minimum duration of a phoneme. The alignment between states and the segments of the phoneme are flexible and adjusted during the training process.
The transition model between states provides a self loop to remain in the current state (slow pronunciation), a step into the next state (normal) or a transition skipping one or more states for fast pronunciations (not shown in the picture).
Joining together the states of different phonemes produces *Markov Chains* for whole words. A speech recognition system works by comparing the probability that the observed sequence of feature vectors was produced by the *Markov Chain* representing a given word to the probabilities of it being produced by the chains for other words in the dictionary.
Especially at the transitions between phonemes, the speech signal is known to depend on the identity of the phonemes to both sides of the current phoneme. The spectrum of the phoneme `L` will look different if surrounded by vowels than if followed by an `Z`. This is why models for phonemes in the context of one or more adjacent phonemes, called *polyphones*, are used as common units to model speech. If the same model is used for several similar *polyphones*, these models are referred to as *allophones*.



Example *Markov Chain* for the word *aisles*

2.2.5 Observation Probabilities

As part of the computation of $P(A|W)$ it is necessary to compute the probability density $f(x|s)$ of a given feature vector x being generated by a single polyphone state s.

When modeling speech with *hidden Markov models (HMM)*, it is assumed that $f(x|s)$ only depends on the current feature vector and state. The probability density can be approximated by a superposition of M Gaussian distributions. If the coefficients in the feature vectors are not correlated, this probability density is computed as:

$$f(x|s) = \sum_{m=1}^{M} c_{sm} \exp -(\sum_{d=1}^{D}(x_d - \mu_{smd})^2/2\sigma_{smd}^2) \tag{2.8}$$

All mean vectors μ_{sm} of the Gaussians for a model s, together with their variances σ_{sm}, will be referred to as the *codebook* for the model. The mixture coefficients c_{sm} of the Gaussians for a model s will be called *weights*.

Several allophones can share one codebook. Models that share a codebook can describe different *allophones* by using different sets of weights. If this is not done, and all models that share one codebook use the same weights, the HMM is called *fully continuous*. If on the other hand there is only a single codebook in the system, and the differences between different models are expressed by different sets of weights only, the resulting HMM is a *semi-continuous HMM*.

Recognizers that do not use different allophones for one phoneme in different contexts are called *context independent*.

Since an understanding of the HMM training algorithm is not required for the work presented in this thesis, it will not be described here. An excellent tutorial is found in [Rabiner, 1989].

2.2.6 Forward Algorithm

The goal of the *forward algorithm* is to compute the probability $P(A|W)$, that is to determine how likely an observed sequence of feature vectors was generated by the *Markov Chain* representing the word sequence W. To that end, for each state, the probability for observing the sequence up to a given feature vector is computed over all possible state sequences leading to that state (figure 2.3).

The observation probability for a feature vector in a *hidden Markov* state s_i is usually assumed to depend only on the current feature vector x_j and the state s_i, but not on any earlier states or inputs. Therefore, the probabilities can be computed by a simple iteration over the input vectors. The resulting algorithm is known as *forward algorithm*.

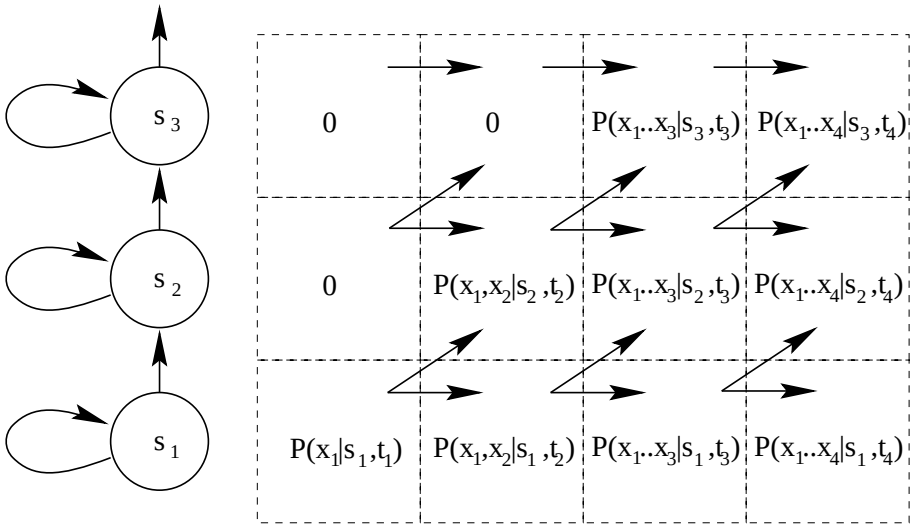


Figure 2.3: Example for the *forward algorithm*.

When processing the first feature vector, the probability for being in the first state of the chain must be one and the probability for being in any other state must be zero, since it is not allowed to start the sentence in the middle of a word:

$$\begin{aligned} P(x_1|s_1,t_1) &= P(x_1|s_1) \\ P(x_1|s_2,t_1) &= 0 \text{ (boundary value)} \\ P(x_1|s_3,t_1) &= 0 \text{ (boundary value)} \end{aligned} \tag{2.9}$$

For all subsequent input vectors x_i, the probability for observing the sequence up to x_i and ending up in state s_j is computed by calculating the sum over the probabilities of all legal predecessor states at t_{i-1}, multiplied by the observation probability for the current feature vector x_i in this state.

$$\begin{aligned} P(x_1 \ldots x_i|s_1,t_i) &= P(x_1 \ldots x_{i-1}|s_1,t_{i-1})P(x_i|s_1) \\ P(x_1 \ldots x_i|s_2,t_i) &= (P(x_1 \ldots x_{i-1}|s_1,t_{i-1}) + P(x_1 \ldots x_{i-i}|s_2,t_{i-1}))P(x_i|s_2) \\ P(x_1 \ldots x_i|s_3,t_i) &= (P(x_1 \ldots x_{i-1}|s_2,t_{i-1}) + P(x_1 \ldots x_{i-i}|s_3,t_{i-1}))P(x_i|s_3) \end{aligned} \tag{2.10}$$

After processing all feature vectors, the total cumulative probability in the final state of the chain is the probability for the observed sequence given this particular *Markov chain*.

For the implementation of the *forward* algorithm, measures must be taken to avoid a floating point underflow: the product over several small probabilities can easily take values below the resolution of the chosen data type. Since the algorithm requires the computation of a sum over probabilities, the resolution for small probabilities cannot easily be increased by using logarithmic probabilities instead.

2.2.7 Viterbi Algorithm

The *Viterbi* algorithm is used to find the most likely sequence of states in a *Markov chain* to produce an observed sequence of feature vectors. The observation probability along the best state sequence through the *Markov chain* can also be used as an approximation for the probability of that chain producing the observed sequence of feature vectors. This estimate can be used to build a simple isolated word recognizer.

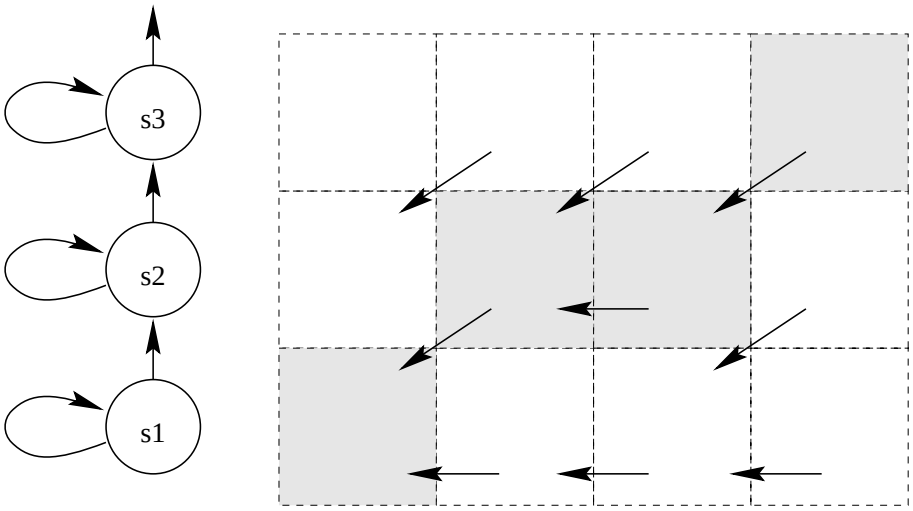


Figure 2.4: Example for the *Viterbi* algorithm. The arrows indicate the best predecessor state for each point in time and each state. The gray fields mark the best path through the chain.

Since the utterance has to start at the beginning, at the time t_1 the only legal state is s_1. Legal successors of s_1 are s_1 and s_2, both of which do not have other legal predecessors at the time. Hence, the best predecessor state of s_1 and s_2 for t_2 is s_1. The probabilities of the corresponding state sequences are $P(x_1|s_1)P(x_2|s_1)$ and $P(x_1|s_1)P(x_2|s_2)$.

At t_3, both s_1 and s_2 are legal predecessors for s_2. For the *Viterbi* algorithm the better one (s_2) is chosen giving $P(x_1|s_1)P(x_2|s_2)P(x_3|s_2)$ as probability for observing x_1, x_2, x_3 along the best state sequence to s_2.

For all states s_j and points in time t_i the observation probability along the most likely state sequence is computed by:

$$P(x_1 \ldots x_i|s_j, t_i) = P(x_i|s_j)) \max(P(x_1 \ldots x_{i-1}|s_j, t_{i-1}), P(x_1 \ldots x_{i-1}|s_{j-1}, t_{i-1})) \quad (2.11)$$

To be able to retrieve the best state sequence, a pointer to the best predecessor is stored for each state and point in time. After processing the last feature vector, the trail starting at the last state of the *Markov chain* can be followed backwards to the first state and frame.

The big advantage of the *Viterbi* algorithm over the *forward* algorithm is that no sums of probabilities need to be computed. Therefore it is possible to simply work with the logarithm of all probabilities, computing sums of logarithmic probabilities rather than products of probabilities. The resulting logarithmic score range stays well within the limitations of most computers. Logarithmic probabilities are very common in speech recognition. In this thesis, the word *score* always refers to a negative logarithmic probability: high scores always mean low probabilities and low scores mean high probabilities.

Since the *Viterbi* algorithm aligns the time slices corresponding to the input feature vectors to the states of the *Markov chain* with respect to the observation probabilities, the process is also referred to as *dynamic time warping (DTW)*.

2.3 Continuous Speech Recognition

Using the *forward* algorithm, it is possible to determine for each word sequence W how likely the signal A was produced by the corresponding *Markov chain*. However, it is impractical to enumerate all possible word sequences to find the sequence that maximizes $P(A|W)$. If the recording is segmented into sections A_i containing exactly one word w, $P(A_i|w)$ can be computed for each section to find the best matching sequence of words in A.

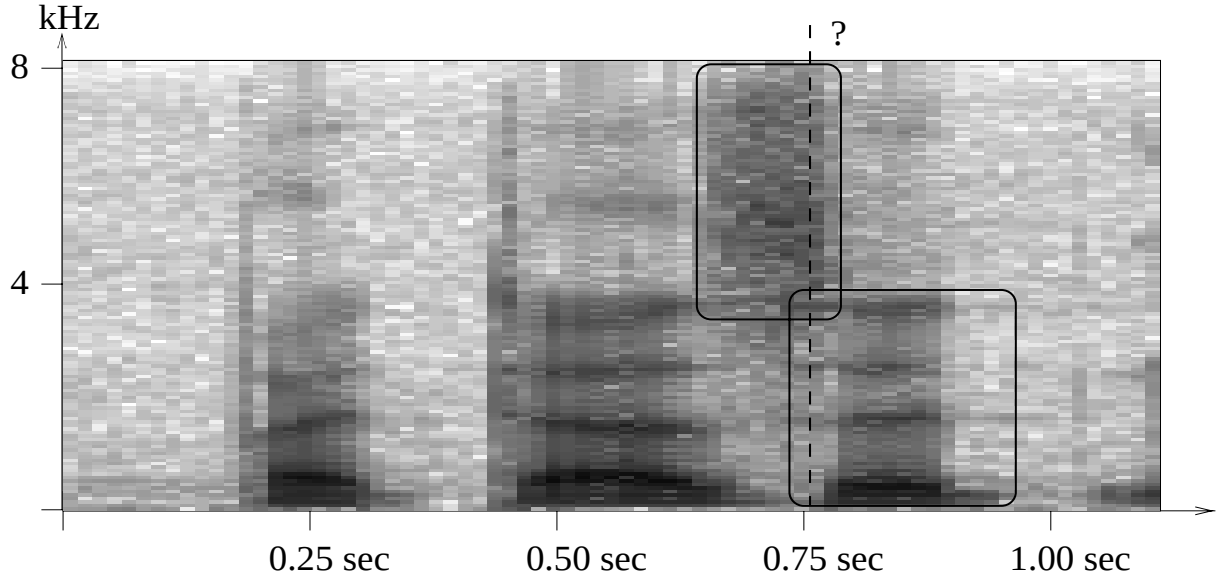


Figure 2.5: The spectrum for the sentence '*but does it*' illustrates that there is no clearly defined word boundary between the final `S` of the word '*does*' and the `I` of the word '*it*'.

Unfortunately, as illustrated in figure 2.5, for continuous speech a correct automatic segmentation cannot always be found.

2.3.1 Time Synchronous Search

A common approach to find the best sentence hypothesis for a sample of continuous speech is the *One-Stage-Dynamic-Time-Warping* algorithm [Sakoe and Chiba, 1978, Sakoe, 1979, Ney, 1984]. It is a generalized *Viterbi* algorithm that combines the segmentation with the recognition process. Within the *Markov chain* for a word, the algorithm finds the probability along the best path through the word using the *Viterbi* algorithm. However, for each frame there are additional possible transitions from all word ends into all word-begin states. While in the normal *Viterbi* algorithm the only legal predecessor of a word-begin state is this same word-begin state, all word-end states are now considered to be possible predecessors.

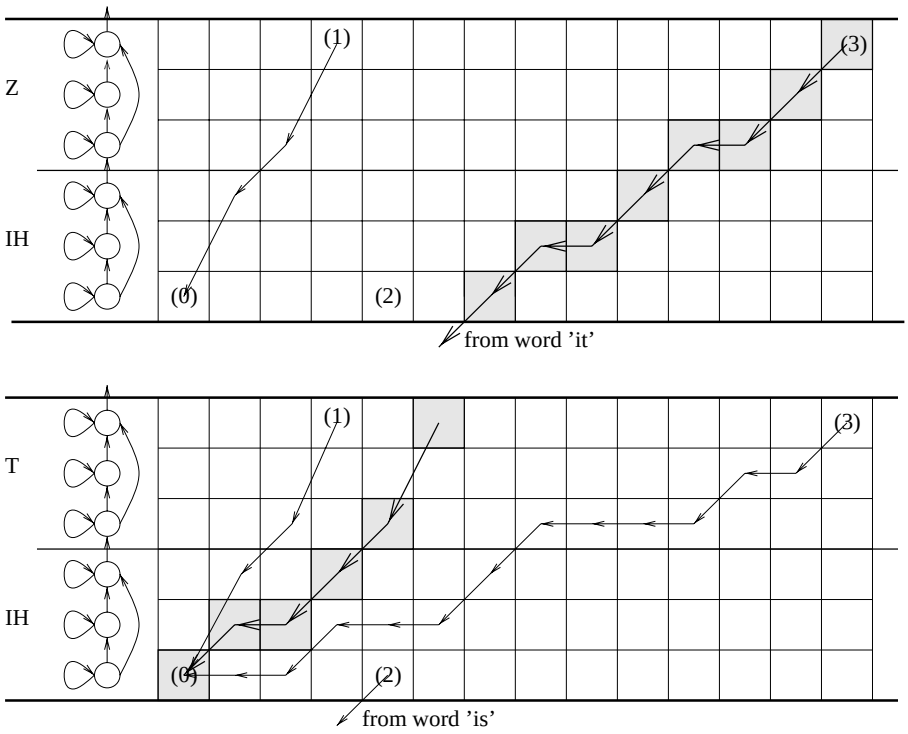


Figure 2.6: Example for the One-Stage-DTW: (0) possible start points for the sentence, (1) the first possible word-end points, (2) the first possible word-transitions, (3) possible end points for the sentence. The best path (gray background) is found by following the pointers backwards.

Figure 2.6 gives an example for two words with two phonemes each: *it* and *is*. From these words sentences like '*it is.*' or '*is it?*' or '*it is it.*' can be built.

All word-begin states in the first frame (0) are possible start points for a sentence. After a few frames, the transition into a word-end state is possible (1). In the next frame (2) all word-begin states must consider these word-end states as legal predecessors. At this point the language model transition probabilities $P(w_{new}|w_{old} = \text{is})$ and $P(w_{new}|w_{old} = \text{it})$ can be included. This means the best predecessor of the word *it* is not necessarily the best predecessor of the word *is*. → 17

Upon reaching the last frame, the algorithm takes the word end with the best score and follows the pointers backwards just like the *Viterbi-Algorithm*. Every time the pointer crosses a word-boundary, it produces one word of the sentence hypothesis that is thus generated in reverse order.

Since the required multiplications can be replaced by sums in the logarithmic space, it is common to use the negative logarithm of the cumulative probabilities. Hence, a large score implies a small probability and a small score a large probability.

To reconstruct the best word sequence, the only required information is which word was the best predecessor to the current word end. Therefore, for each word end, the following information is stored: which was the best predecessor word and at what frame was the transition from this predecessor into the current word made. This structure, marked as long arrows in figure 2.7, is called the *backtrace*.

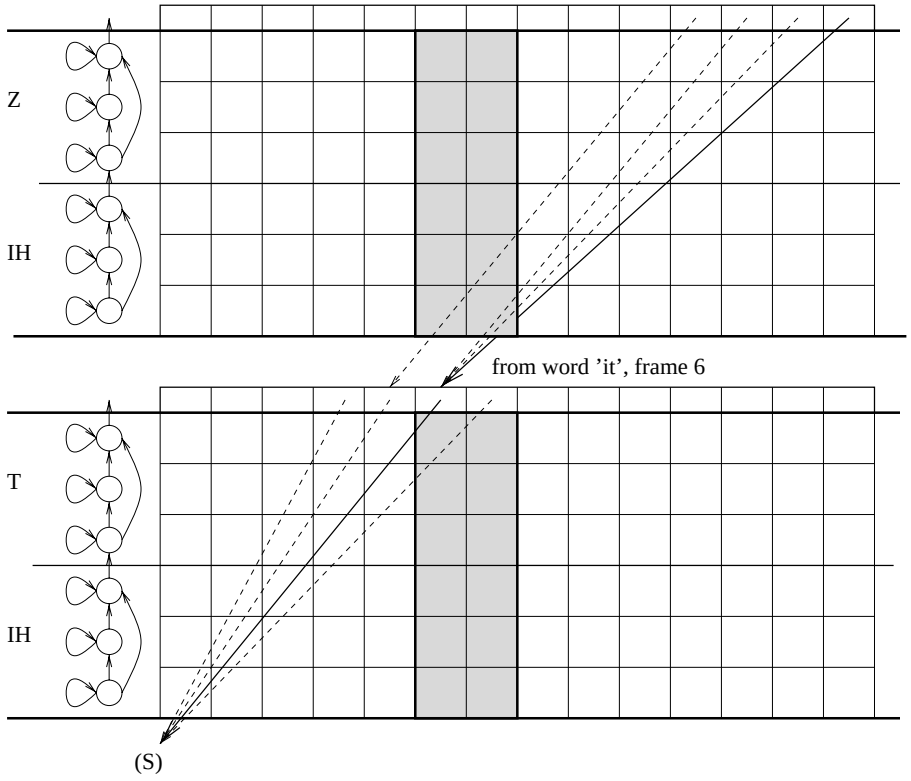


Figure 2.7: Computing the One Stage DTW using 2 frames and a *backtrace*. The *backtrace*, represented as flat rectangles above the word models, contains a pointer to the best predecessor word and starting point for each word end and each frame.

Note that if no language model is used, the best predecessor word for all words starting in one frame is the word with the best score in its word-end state for the previous frame. In this case the backtrace looks the same for all words and could be replaced by a single back-pointer per frame. If, however, the best predecessor depends on the identity of the word because of a bigram language model, one back-pointer has to be stored for each frame and each word end.[3]

[3] Using trigram language models complicates matters even more and would exceed the scope of

The information that is to be stored in the backtrace upon reaching the last state of a word could be obtained by following the back-pointers within that word from state to state. However, it is much more efficient to maintain the information of the word entry frame and the predecessor word in the data structure of each state for the current frame. In the next frame, each within-word state inherits this information from its best predecessor state. Thus, no further information about the exact path within the words needs to be stored. The memory requirement is reduced to the memory used for the backtrace plus the contents of all states for the current and the previous frame. The incomplete sentence hypothesis that is under construction in each of the states of the *Markov chain* is called a partial hypothesis. Many partial hypotheses have cumulative scores that are so bad compared to other hypotheses that they are unlikely to be the best choice at the end of the sentence. These partial hypotheses can be *pruned* from the search space. Since pruning is a rather vital speed issue, and because several pruning strategies have been investigated in the work for this thesis, more detailed information and experiments can be found in chapter 4.2.1. → 39

N-best Hypotheses

For many applications, knowing the best word sequence is not enough. Instead, it might be beneficial to know what the second best sequence looks like and whether other sequences get a similar score.

Such lists of hypotheses can be re-ordered by applying expensive acoustic models or long-range language models that cannot be used during the main recognition process because they require the unavailable right context of a word [Austin et al., 1992]. Another application of N-best lists is the fine-tuning of parameters such as the language-model weight [Schwartz et al., 1992].

Also, if the second best hypothesis has the opposite meaning of the first, it may be a good idea to reject the recognition (e.g. *do/don't delete files*).

To build N-best lists based on a time synchronous search, there are a number of common algorithms. The most accurate but also slowest approach is the *exact N-best* algorithm. For each state it uses an N-deep stack of partial hypotheses sorted by their cumulative score. To build the new stack for a state at the time t_i, all partial hypotheses from all predecessor states at time t_{i-1} are considered. From those, the N-best partial hypotheses that differ in the associated word sequence are chosen for the new state (figure 2.8). The first state of a word can therefore have the final states of N differing words as predecessors. At the end of the utterance, the N best hypotheses can be extracted by following the N best of all partial hypotheses in the stacks of all word-end states.

For more detailed information about algorithms to find the N-best hypotheses refer to [Schwartz and Chow, 1990, Schwartz and Austin, 1991].

a fundamentals chapter. This problem will be addressed in more detail in chapter 5.1.2 (→ p. 59).

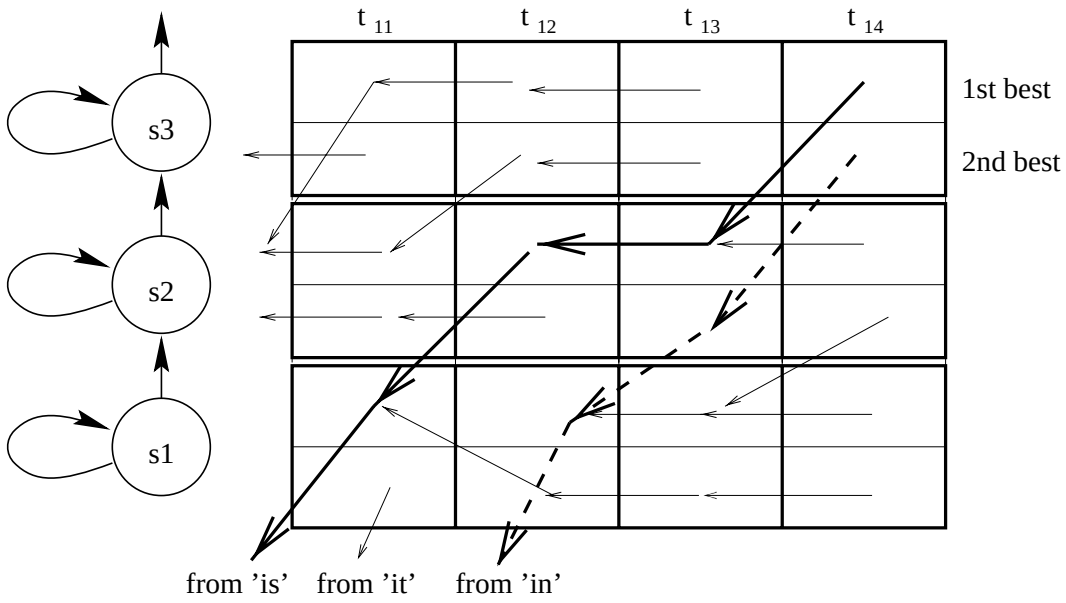


Figure 2.8: Example for the Computation of the exact N-best algorithm (N=2). For each state, there is a stack of possible partial hypotheses representing differing word sequences.

2.3.2 Depth First Search

All search algorithms mentioned so far are time synchronous, and are therefore basically breadth first searches. The depth first search, however, keeps all partial hypotheses on a stack and expands only the hypothesis that looks most promising at
→ 125 the moment [Paul, 1992b, Paul and Neioglu, 1993]. Once this hypothesis reaches the final frame of the utterance, the best overall hypothesis is available. The algorithm can be set up to either stop then, or to continue until the second best hypothesis reaches the last frame, and so on until the resulting N-best list is long enough for the application in question. The main problem with the depth first search is knowing which partial hypothesis is more promising, because the current scores for different partial hypotheses can be based on a different number of frames. Also, the efficient organization of the hypotheses in the stack can be quite demanding. Therefore, depth first searches are not often the method of choice for the first pass
→ 120 of a decoder. They are sometimes found as a second pass or used to combine the results of several search passes with new information sources [Alleva et al., 1993].

Chapter 3

Test Environment

The nice thing about standards is
that there are so many of them to choose from.
– Andrew S. Tanenbaum

This chapter gives an outline of the environment used for the experiments described in later chapters. There are three main sections: The first section gives an overview of the the JANUS recognition toolkit that has used as development toolkit throughout this work. The *North American Business News (NAB)* database used for training, test, and evaluation of the recognizers and algorithms is presented in the second section. Finally, the actual NAB recognition system used as baseline for most experiments is described.

3.1 The JANUS Recognition Toolkit

Strictly speaking, JANUS is a speech-to-speech translation system. However, the speech recognition component used for JANUS was originally also called JANUS.

The JANUS speech recognition engine has been around since before 1991, though by now there is probably not a single component of the original system that has not been been rewritten several times or completely removed. After a complete re-implementation in 1995, the JANUS recognition engine was renamed to JRTk (JANUS-Recognition-Toolkit) to avoid further confusion and to reflect its new capabilities.

Since 1991, the author has contributed to all versions of JANUS, especially by working on the search but also on the runtime performance of other algorithms such as the computation of observation probabilities. However, this is probably the place to point out that JRTk is a joint effort of many dedicated researchers. Without their contribution to the recognition system, most parts of this thesis would never have been written.

The JRTk User Interface

JRTk is not just a speech recognizer, but a speech toolkit with a programming interface. The programming interface is realized as an integrated tcl interpreter [Welch, 1995], used to run scripts from which the user can create the objects that make up the recognizer, such as preprocessing objects, codebooks and a search.

The main advantage of a programmable toolkit is that many experiments on new techniques can be performed without changing a single line of the JRTk source code. The main disadvantage is that the large flexibility often reduces the efficiency and makes it difficult to automatically detect errors in scripts of novice users. A library of standard scripts for common tasks is used to minimize those problems.

By combining the graphical interface of tcl-tk with JRTk, many of the internal objects of JRTk can be visualized and used to quickly create prototypes for new
→ 108 speech applications.

Observation Probabilities in JRTk

JRTk contains many different variants for the computation of observation probabili-
→ 23 ties, ranging from completely semi-continuous, to tied mixture systems (e.g senones as described in [Hwang and Huang, 1992]), and to fully continuous density HMMs. The Gaussians can be modeled using no covariances, radial covariances, diagonal covariances, or full covariance matrices. Acoustic modeling using neural nets, alone or in combination with HMMs is also possible [Fritsch et al., 1997].

However, most of the systems realized with JRTk use the following architecture: each *polyphone* is divided into three states. These sub-polyphones are clustered to build 2,000 to 5,000 sub-allophones [Finke and Rogina, 1997]. To model each of the sub-allophones, a codebook containing 16 to 48 Gaussians with diagonal covariances is used. Further context information can be introduced by using different sets of weights to differentiate between sub-allophones that share a codebook.

Details on the fast computation of observation probabilities in JRTk are introduced throughout the following chapters.

Search in JRTk

JRTk uses a multi-pass search strategy. The underlying idea is to use a first search pass to restrict the search space for the second pass, which uses better but more expensive algorithms to find the best word sequence. More details about this search
→ 56 algorithm are given in chapter 5.1.

3.2 The *North American Business News* Task

All algorithms presented in this thesis have been developed for and tested on the *North-American-Business-News (NAB)* dictation scenario.

The NAB task is the successor of the better known *Wall Street Journal (WSJ)* task. Both tasks are recordings of read newspaper articles, collected for the evaluations organized by the *Defense Advanced Research Agency*, DARPA, to compare the performance of speech recognition systems.

The evaluation conditions clearly divide the available material into training, test and evaluation data and they define fixed word lists and language models for the so called *hub* and *spoke* tests. All NAB results presented in this thesis have been computed on data used for the last official evaluation of NAB dictation systems in November 1994.

For most experiments in this thesis, the conditions of the main (P0) test were used: For the P0 test, the recognizer could be trained with all available data, and tested using a vocabulary size of 65,000 words. The full test set for the P0 test consists of 316 utterances covering 10 male and 10 female speakers. The total length of the test set is about 53 minutes, or 8,186 words. All P0 recordings were made in a quiet office environment using a high quality close speaking microphone.

The recognition results of the competing recognizers in the 1994 evaluation ranged between 7.2% and 22.8% *Word Error Rate*. Comparing more recent results on the same evaluation set can be misleading as some recognizers have since been tuned to this set.

Many of the experiments presented in this thesis iterate over several parameters. To conserve time and reduce the memory requirements, some of these experiments were performed using a subset with only 7,500 of the 65,000 words of the full vocabulary. This subset was built to contain the same number of out of vocabulary words on the test set as the 65,000 word vocabulary. The normal out of vocabulary rate for such a small vocabulary would be over 5%, introducing new recognition errors, thus making it harder to compare the results to the large vocabulary experiments. For all important experiments and conclusions, comparative results on the full vocabulary are provided.

Also, a subset of 50 sentences from the female test set covering all 10 speakers was chosen for most experiments. Additional experiments were performed to show the influence of the size of the test set on the confidence intervals of the results. → 96

Robustness and portability studies have been performed on further data sets taken from other subsets of the NAB evaluation (so called *spoke conditions*) and from recordings of German, English, and Japanese spontaneous speech. Some results are also available on the data of a LVCSR dictation system for Japanese.

More details about the NAB task and evaluation results on the P0 and spoke tests are presented in the proceedings of the DARPA-workshop [ARPA Workshop, 1995].

3.3 The Baseline NAB Recognizer

The baseline recognition system used for the experiments on NAB was trained by Ivica Rogina [Rogina, 1997] with the only goal being to minimize the error rate on the 1994 P0 evaluation data. Robustness and recognition speed were no criteria at any point in the development of the baseline system. The error rates on the NAB 1994 P0 evaluation data achieved by the system that served as baseline for all NAB experiments in this thesis were around 9%.[1] Under evaluation conditions, this system ran at 200 times real-time. Setups and results for the systems of other
→ 115 groups in this evaluation are given in the appendix.

Preprocessing

→ 18 The speech signal is sampled with a resolution of 16 bit at 16 kHz. Using 16 millisecond long, overlapping windows, a vector of 16 *Mel-scale* coefficients is computed for every 10 milliseconds of speech input. The mean of all vectors in one utterance is subtracted from each vector. This is the only step in the preprocessing for which the whole utterance is required.

Using a window of seven consecutive vectors a *'super-vector'* is constructed. This *super-vector* is mapped to the 48-coefficient feature vector using a matrix multiplication. The matrix for this step is determined by means of a *linear discriminant analysis* (LDA) to minimize the within class scatter and to maximize the between class scatter of the data assigned to the allophone segments of the acoustic model [Fukunaga, 1990].

Acoustic Models

→ 23 The system used in this thesis contains 3,000 context dependent codebooks with 32 Gaussians in a 48 dimensional feature space. For each codebook there is exactly one set of weights defining the model for one allophone segment (fully continuous density HMM). The maximum context width used is two phonemes in each direction, resulting in quinphone models. Two separate systems were trained for male and female speakers.

Language Models

→ 17 For all NAB experiments the official 64K trigram language model `t95+60k1` with 8,814,128 trigrams and 7,454,368 back-off bigrams was used. This language model was built on a text corpus of more than 300,000,000 words and is provided as part of the NAB evaluation materials.

[1] Current error rates of an improved system with larger codebooks and speaker adaptation are around 7%. Many preliminary experiments for this thesis were based on an older system with an error rate of 23% and have been repeated with the 9% system.

Chapter 4

Speed Issues in Speech Recognition

Any sufficiently advanced technology
is indistinguishable from magic.
– Arthur C. Clarke

This chapter gives an overview of some techniques that have been used in the past to speed up recognition systems. Algorithms that were developed by the author or are very closely related to the thesis contribution are presented in chapter 5. → 55

There are four main sections in this chapter:

Language Models: While computing the language model probabilities is no speed issue by itself, using language models considerably complicates the recognition process. This section presents two common approaches used to reduce the impact of the language models on the recognition time. → 36

Search: Finding the best word sequence in a large search space is a computationally demanding task. Methods such as pruning the search space and using a tree organized lexicon are commonly used to reduce this effort. Often, improvements in the search structure also reduce the time required for other subtasks. → 39

Observation Probabilities: Most LVCSR systems spend 60% to 90% of the total CPU-time on the computation of observation probabilities with mixtures of Gaussians, depending on the vocabulary size and the complexity of the acoustic models used. This section presents some faster algorithms to compute these observation probabilities and an alternative approach using neural nets. → 45

Parallel Computing and Hardware: If the algorithmic approaches are not enough, it is always possible to resort to faster machines or special purpose hardware. The last section in this chapter describes some of these hardware related approaches. → 49

4.1 Language Models

The bigram and trigram probabilities used for the language modeling are usually pre-computed before the recognition run. Computing the transition score then corresponds to one or two table lookups. This it not an expensive operation.

Whenever language models are used, the best predecessor word can be a different word end for every word-begin state. Therefore, for the word-begin states of all words in the vocabulary (e.g. 65,000) the best predecessor has to be found among all possible word ends. This has to be done for every frame, about 100 times per second of input speech. Even if the number of active word-end states is reduced
→ 39 to about 200 states by pruning strategies, evaluating the remaining 1,300,000,000 word transitions per second still presents a problem.

4.1.1 Smoothed Language Models

When generating statistical language models from large corpora, most of the possible n-grams are not observed in the training text or are not frequent enough for a robust probability estimation.

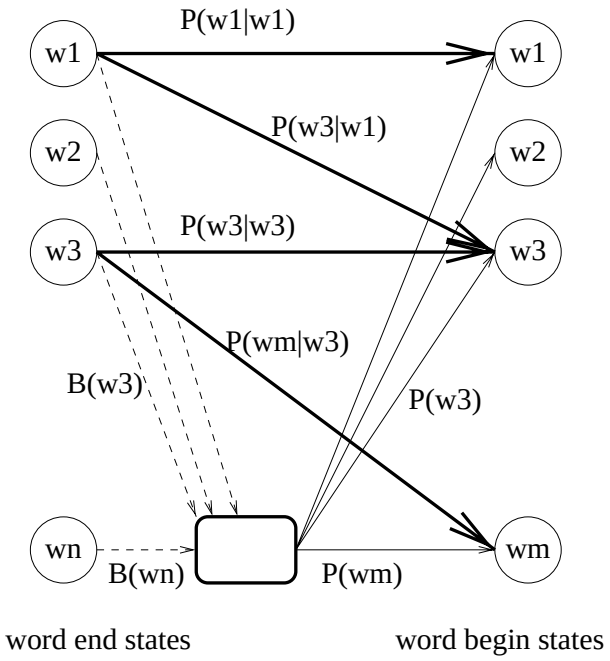


Figure 4.1: For back-off bigram models, the probability $P(w_m|w_n)$ for unseen bigrams can be divided into two terms $P(w_m|w_n) = P(w_m)B(w_n)$. The best predecessor word for all unseen word transitions is the same for all words. Only word transitions that have been observed often enough to provide a robust probability estimation depending on both words have to be explicitly computed (fat arrows).

For the *North-American-Business-News* corpus, the language model contains probabilities for 60,000 unigrams but only for $5 * 10^6$ bigrams, though $60,000^2 = 3.6 * 10^9$

bigrams are technically possible. The probabilities for all remaining bigrams (99.9%) have to be estimated using back-off strategies [Katz, 1987].

For the computational effort in speech recognition, the most interesting aspect of these back-off strategies is that for unseen bigrams, $P(w_2|w_1) = P(w_2)B(w_1)$ is composed of two terms of which one depends only on the predecessor word and the other only on the successor. For all word transitions with unseen bigrams the best predecessor is the same word. The effort for finding this predecessor to compute all *unseen* transitions grows only with $O(n)$, where n is vocabulary size. The effort for finding the best predecessor for all transitions with explicit bigram information is still large as it grows with the number of observed bigrams in the training corpus, but much smaller than $O(n^2)$;

4.1.2 Delayed Bigrams

When using *Delayed Bigrams*, the language model information is included after computing the transition into a successor word, for instance upon computing the transition into the last phoneme of the successor word.

At the transition into the last phoneme of a word w_{new} the transition probability into w_{new} is included. To that end, the entry point into w_{new} that is stored in the partial hypothesis is used to determine which of the words ending at the entry point would be the best predecessor of w_{new} when including the transition probability $P(w_{new}|w_{old})$;

Since many words are pruned from the search space before reaching their last phoneme, this approach avoids the rather costly procedure of finding the best predecessor for these words. Also, this method agrees with search strategies that organize the search lexicon into a phonetic tree where the identity of the successor is not available until a tree leaf, usually the last phoneme of the word, is reached. → 41

The most serious disadvantage of *Delayed Bigrams* is that the entry point for the successor word is determined without any language model information. This lack of information causes segmentation and pruning errors depending on the importance of the language model. For the NAB experiments in this thesis this leads to a relative increase in error rate of about 12%.

A second search pass can be used to recover these errors if the influence of the language model is only moderate. → 59

The generation of segmentation errors is illustrated in figure 4.2. Suppose that at the time t_1 the best word end is is w_1, and at the time t_2 the best and possibly only word end is w_2. A couple of frames after t_2, but prior to the transition into the last phoneme, the two resulting partial hypotheses are both legal predecessors to the same state. The best predecessor is chosen only according to the current scores in the partial hypotheses, without respect to language model information. The word entry point is set to t_2, though using the language model the best predecessor and

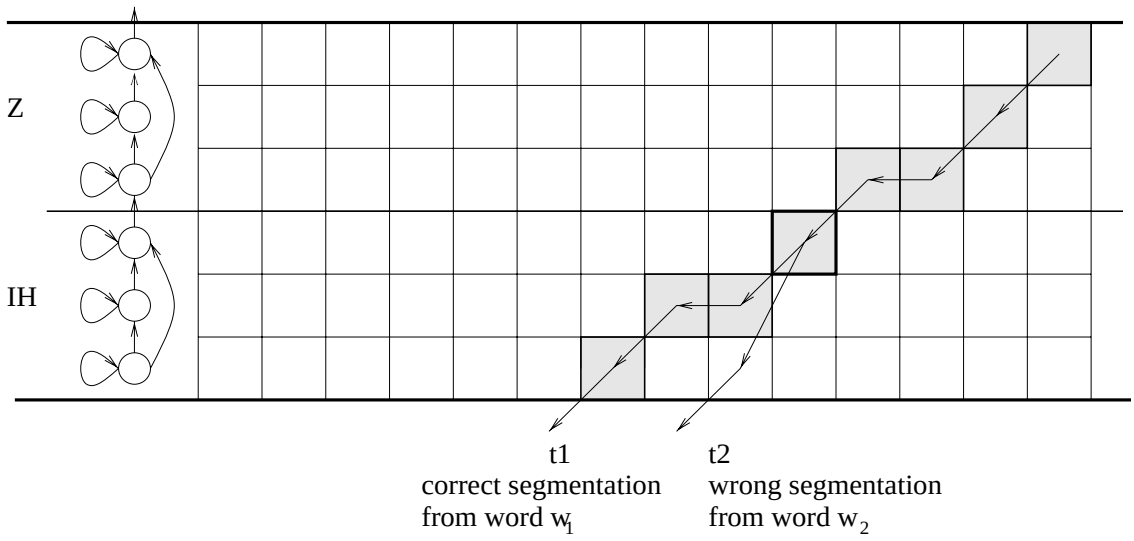


Figure 4.2: Example for a segmentation error caused by the use of delayed bigrams.

entry point word may have been t_1. Since the segmentation is wrong, the predecessor word can also be incorrect. Now consider the case of a language model that has a zero probability for the transition from w_2 to the current word. In this case the probability of the currently best and quite possibly only active hypothesis suddenly drops to zero resulting in an empty recognition output. This is the reason why *Delayed Bigrams* should not be used with restrictive language models.

4.2 Search

For large vocabularies, the search space grows quickly. When using a dictionary with 65,000 words that each have average of about 5 phonemes with three states, the full search space contains close to 1,000,000 partial hypotheses at each point in time. Using even simple pruning strategies reduces this number significantly.

However, pruning tends to be more efficient in eliminating partial hypotheses towards the end of the words. At the beginning of a word there is no acoustic evidence how well the word will match. Therefore, many word-begin states are active. When the search lexicon is organized into a tree the number of word-begin states becomes significantly lower.

Multi-pass search strategies can further help to reduce the total recognition time as illustrated with the *forward-backward-search* algorithm and the section on word graphs.

4.2.1 Pruning the Search Space

Many partial hypotheses have a score that is so bad compared to other hypotheses, that they are unlikely to be the best choice at the end of the sentence. These partial hypotheses can be *pruned* from the search space.

Some of the more advanced pruning strategies that have been investigated in the work for this thesis are described in in section 5.1.4. → 63

Score related thresholds

The easiest way to prune the search space is to determine the score of the best partial hypothesis for a frame and then remove all partial hypotheses from the search space that have a cumulative score that is worse by a specified value. If a partial hypothesis is removed from the search space, the corresponding state is said to be *inactive*. It can be reactivated by computing a transition from an *active* state into it.

All remaining partial hypotheses now have scores in the range determined by value used for the pruning. This value is called a *beam*. If the acoustic and language models describe the speech signal well, the distance between the best and the second best hypothesis will be large, and only a few states will remain active. In difficult sections, the number of active states increases.

One of the drawbacks of pruning with beams is that good values for pruning thresholds depend on the modeling used for the recognizer. The likelihood of removing the correct hypothesis and thus introducing search errors increases for small beams, as shown in the example in figure 4.3. Values that yield acceptable error rates for a given application are usually determined by trial and error.

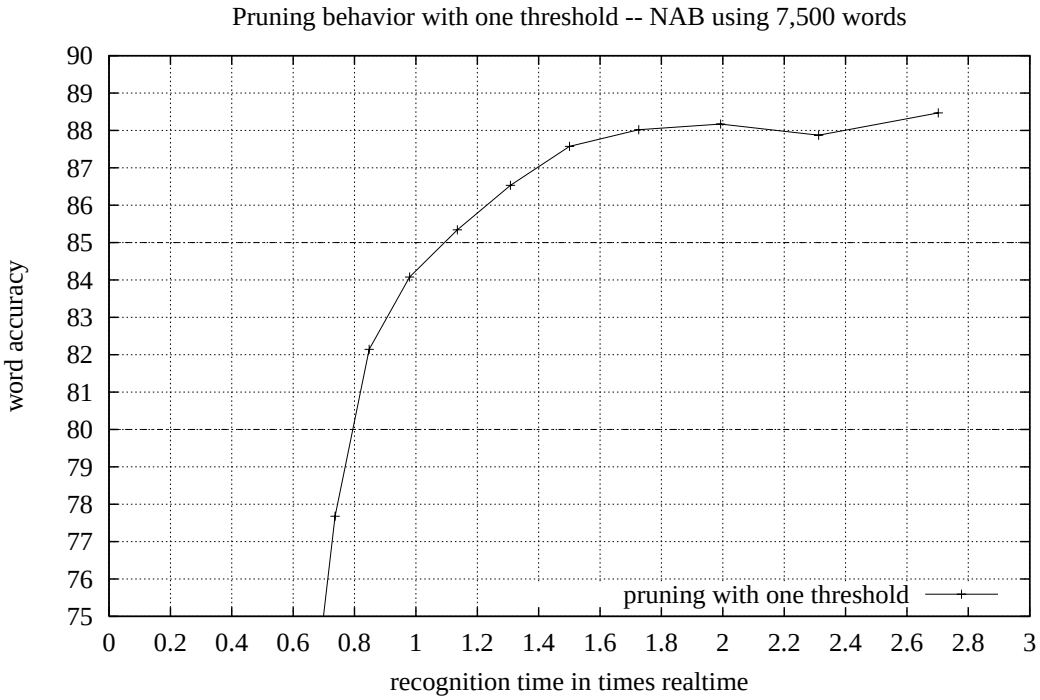


Figure 4.3: Pruning with a score related threshold for an NAB experiment (50 female sentences, reduced vocabulary). The RT=0.74 WA=77.7 value corresponds to a pruning beam-width of 170. The RT=1.73 WA=88.0 value corresponds to a pruning beam of 230.

Special care must be taken if the score increase is not smooth at certain points of the search. At the transition across word boundaries for instance, the word transition probability is included. The word-begin states that have a word end as a predecessor are pruned against the state with the best overall partial hypothesis which could very well be a word-end state. If the bigram penalty $-log(P(w_{new}|w_{old}))$ that is added during the word transition exceeds the beam width, the partial hypothesis in the new word will be always be pruned right after the transition. This can lead to a significant loss in word accuracy for small beams.

Also, If there is a strong mismatch between training and test conditions, most hypotheses will get a bad score and no single hypothesis scores much better than all others. In this event too many hypotheses will remain *active* and the search time spent on this section increases significantly, though the recognition result is still likely to be wrong. This can be a serious problem if a certain response time is required for a given application.

Throughout the following chapters there are graphs such as in figure 4.3 that plot the word accuracy over the recognition time. Unless stated differently in the description of the experiment, data points within one line belong to the same experiment and differ only by the setting of the pruning thresholds. The goal of these plots is to show that the cost in terms of word accuracy loss for a given speed-up is not larger

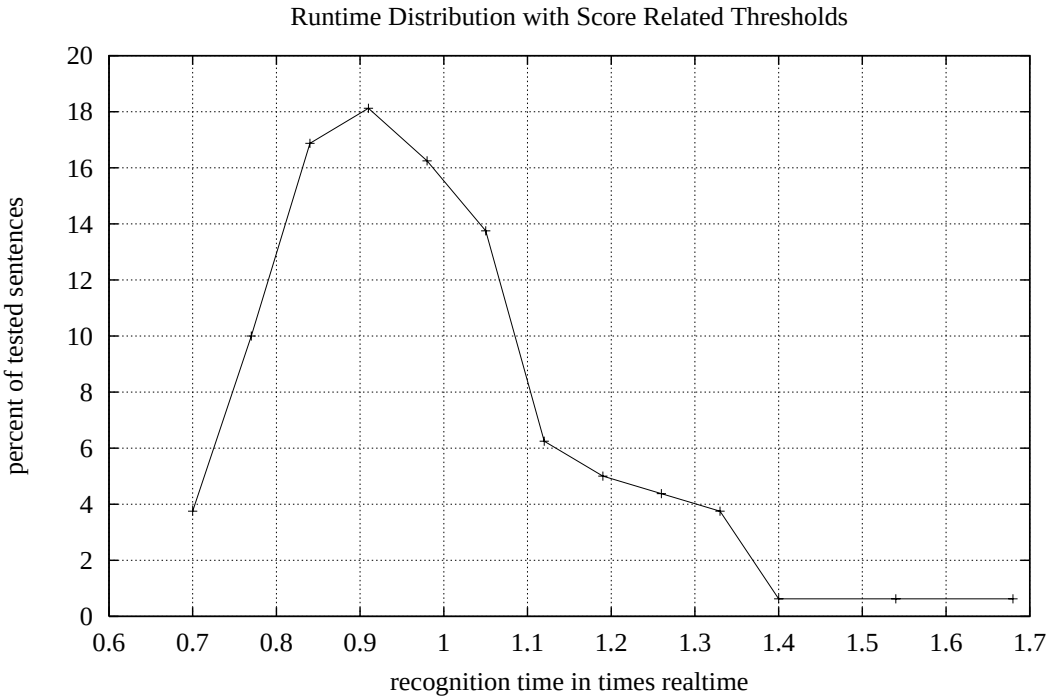


Figure 4.4: Pruning with a score related threshold for an NAB experiment (160 female sentences, reduced vocabulary). The recognition time for one sentence differs between 0.7 and 1.7 RT.

than it would have been if the speed-up had been achieved by reducing the pruning thresholds.

4.2.2 Tree Search

Many words in the dictionary start with the same phoneme. For all copies of that phoneme, the computation of the *Viterbi* algorithm is largely identical.

By using a single phoneme copy for all words starting with the same phoneme, the search lexicon is restructured to form a set of trees. For a context independent system the tree structured lexicon has about 50 tree roots (one for each monophone) compared to 65,000 word-begin phonemes in the linear lexicon [Gopalakrishnan et al., 1995]. Trees for context dependent systems, where only words that start with the same allophone share a tree root, are considerably less dense. For the NAB recognition system used in this thesis, 1,000 different word-begin allophones were used. Towards the word ends, the tree becomes less and less compact. The reduction from 65,000 to 1,000 word-begin roots still leads to a significant speed-up, because the pruning is least efficient during these first states of a word.

When using a tree search, the identity of the successor word is not known at the transition of a word end (a leaf in the tree) into a new word (a tree root). Since the

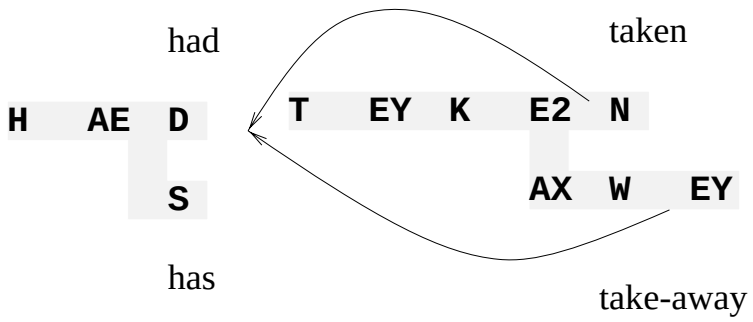


Figure 4.5: Example for *Delayed Bigrams* in a tree search.

successor word is not known, the language model cannot be included at this point. A number of approaches have been proposed to integrate the language model into a tree search [Steinbiss et al., 1994, Alleva et al., 1997].

1. The language model probability is included upon computing the transition into the last phoneme of the successor word, where the identity of the word is always known (*Delayed Bigrams*). The problems of this approach have already
→ 37 been pointed out in section 4.1.2.

2. To avoid the segmentation problems caused by choosing the wrong predecessor due to *Delayed Bigrams*, a new tree copy is allocated for each frame where new words can start [Ney et al., 1992].

3. For each active predecessor word at a word transition, a tree copy is allocated. This way, the actual choice of the best predecessor word is delayed until the language model information is available.

Tree Copies for Start Frames

→ 37 The segmentation errors described in section 4.1.2 can be avoided by creating a tree copy for each point in time where word transitions are possible. This way, paths from different entry points are not merged into one state. Only after the transition into the last phoneme of the word, when the language model has been included, all trees for that word are merged again.

While maintaining tree copies can be tedious, the total effort is still smaller than the effort of a linear search provided that a good pruning strategy is used. Since no systematic error is made in this search strategy, no second recognition pass is required.

Tree Copies for Predecessor Words

A less common approach is to create a tree copy for all active predecessor words. This approach solves the segmentation problem because the segmentation errors

are caused by choosing the wrong predecessor at a time where no language model information is available. Due to pruning, the number of active predecessor words is usually small (about 50 words), putting an upper limit on the increase in search effort.

4.2.3 Forward-Backward Search

If the beams in the beam search are too small, important sentence hypotheses are lost. If the system could use the outlook for a partial hypothesis before deciding to prune it, the likelihood of an error would be reduced.

An estimate of how well a partial hypothesis will do when reaching the end of the utterance would help most. In this case all but the best few hypotheses could be pruned.

If an expensive technique is to be used during the recognition, such as detailed acoustic models or exact N-best searches, this estimate can be produced by means of a preliminary recognition run. The resulting recognizer has multiple search passes, each pass restricting the search space for the next pass.

To that end, it is possible to first compute a *Viterbi*-search backwards, starting at the last frame of the utterance and in the last state of all words. The *backtrace* of this search pass contains the best cumulative score between the end of the sentence and each word entry point for each frame, $\beta(t, w_y)$.

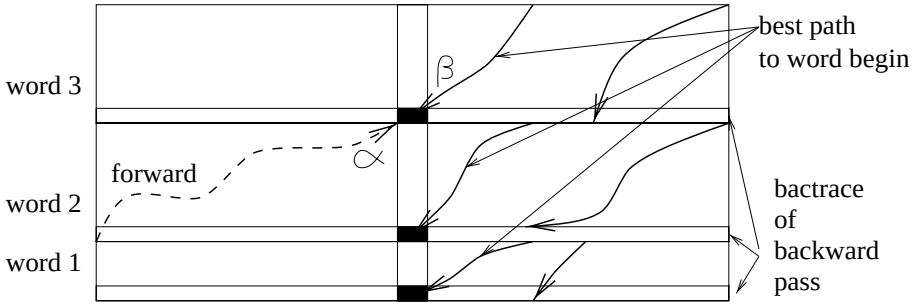


Figure 4.6: Forward-Backward search: at the word transition in the second (forward) pass the outlook of each partial hypothesis is known for all possible word transitions.

The next pass is a forward-oriented search during which the best cumulative score $\alpha(t-1, w_x)$ from the beginning of the utterance to all word ends in a frame is computed. Together with the information from the backtrace of the backward pass, $\alpha(t-1, w_x)\beta(t, w_y)P(w_y|w_x)$ gives the best score this partial hypothesis will have when continued to the end of the utterance along the best possible path. Using this information the pruning beams can be very tight without losing the globally best hypothesis.

One problem with the forward-backward strategy is related to the statistical language models which are built to provide an estimate for $P(w_3 = x|w_1 = y, w_2 = z)$, while during the backward pass, $P(w_1 = y|w_3 = x, w_2 = z)$ is required.

Also, before the search direction can be changed, the whole utterance has to be available. Pipelining with a granularity below the utterance level and run-on recognition are not possible with this approach.

→ 107

The forward-backward search is described in [Schwartz and Austin, 1991]. Some information about the BBN system in which it was first used are found in the Appendix. A similar approach for a multi-pass search is also mentioned in [Murveit et al., 1993].

→ 116

4.2.4 Word Graphs

Word graphs can be used to direct the recognition process of a second search pass, for example with more expensive acoustic models. As the number of word transitions is very limited, highly complex language models can be applied during a search along a word graph.

Also, for many applications a list of likely hypotheses or a word graph that efficiently represents many alternative hypotheses is more useful than the single best-scoring hypothesis.

Alternatives from a word graph can be used in repair dialogues to correct recognition errors [McNair and Waibel, 1994]. For database queries or speech translation systems, the parser that processes the speech input can parse the word graph rather than a single word sequence and thus find a word sequence that is best based on a combination of both the speech recognition knowledge sources and the parsers own representation of the world.

A very efficient algorithm to build a word graph is to simply reuse the backtrace of the search as word graph. To that end, only the information known about the word entry points for each word is used, while the information about the correct predecessor word is ignored. All words ending at the word-entry point of a word are its predecessors for the word graph. Each word in the graph gets a score defined by the score of the partial hypothesis at the end of this word (found in the backtrace) minus the score of the partial hypothesis of its original predecessor.

→ 28

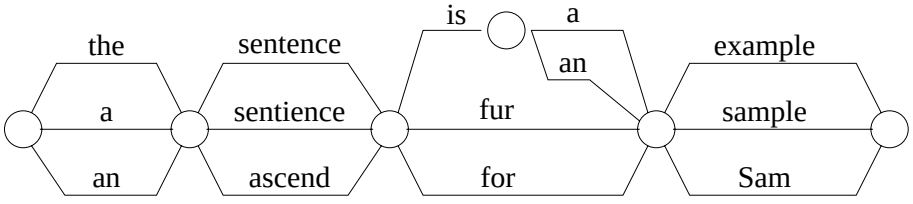


Figure 4.7: A little word graph (81 hypotheses).

If the resulting word graphs are to large for the intended application, they can be pruned as follows: for each word in the word graph, the score of the best hypothesis through this word is computed, including all available language model information. Subsequently, all words that only contribute to hypotheses with scores larger than a given threshold are removed from the word graph.

4.3 Observation Probabilities

To model continuous speech with context dependent phone models, it has become a common approach to use *hidden Markov models* with a large number of codebooks to estimate the observation probabilities. In the case of a system with 3,000 context dependent codebooks with 32 Gaussians in 48 dimensional space, 9,216,000 multiplications are required to compute all observation probabilities for a given frame (assuming diagonal covariances). At a frame-rate of 100 frames per second, this requires 921,600,000 multiplications per second for real time performance. Because the system is not supposed to run only on the few existing teraflop computers, more efficient algorithms for this task have to be developed. → 23

Some of the algorithms used to efficiently compute weighted vector distances were not originally developed for speech recognition tasks but for image and speech compression. The *Nearest Neighbor Approximation* is a typical example for these algorithms.

4.3.1 Nearest Neighbor Approximation

As an approximation for the observation probability it is assumed that

$$\begin{aligned} f(x|s) &= \sum_{m=1}^{M} c_{sm} \exp\left(-\sum_{d=1}^{D}(x_d - \mu_{smd})^2/2\sigma_{smd}^2\right) \\ &\approx \max_{m=1}^{M} c_{sm} \exp\left(-\sum_{d=1}^{D}(x_d - \mu_{smd})^2/2\sigma_{smd}^2\right) \end{aligned} \tag{4.1}$$

This means that only the Gaussian in the codebook that has the smallest *Mahalanobis* distance to the input vector is used in calculating the observation probability.

Since the weighted distances of the Gaussians in a codebook differ by large amounts in high dimensional space, and because this difference is emphasized by the exponential function, the contribution of the remaining Gaussians is often insignificant.

By using only the nearest neighbor for the estimation of the observation probability, the computational effort can be reduced in two ways [Bei and Gray, 1985, Fissore et al., 1993]:

1. When computing the distance to a Gaussian, the partial distance can exceed the currently smallest distance before reaching the last coefficient. Since the actual distance to any vector that is not the nearest neighbor is not used in the score computation, there is no need to complete the distance computation.

2. Because no sum over the weighted distances has to be computed, the computation can be kept completely in the logarithmic space.

Experiments

For systems in a high dimensional feature space, the difference between the distances to the Gaussians within one codebook can be considerable. The contribution of all but the nearest neighbor to the score and the resulting loss in *word accuracy* by using the nearest neighbor approximation is insignificant.

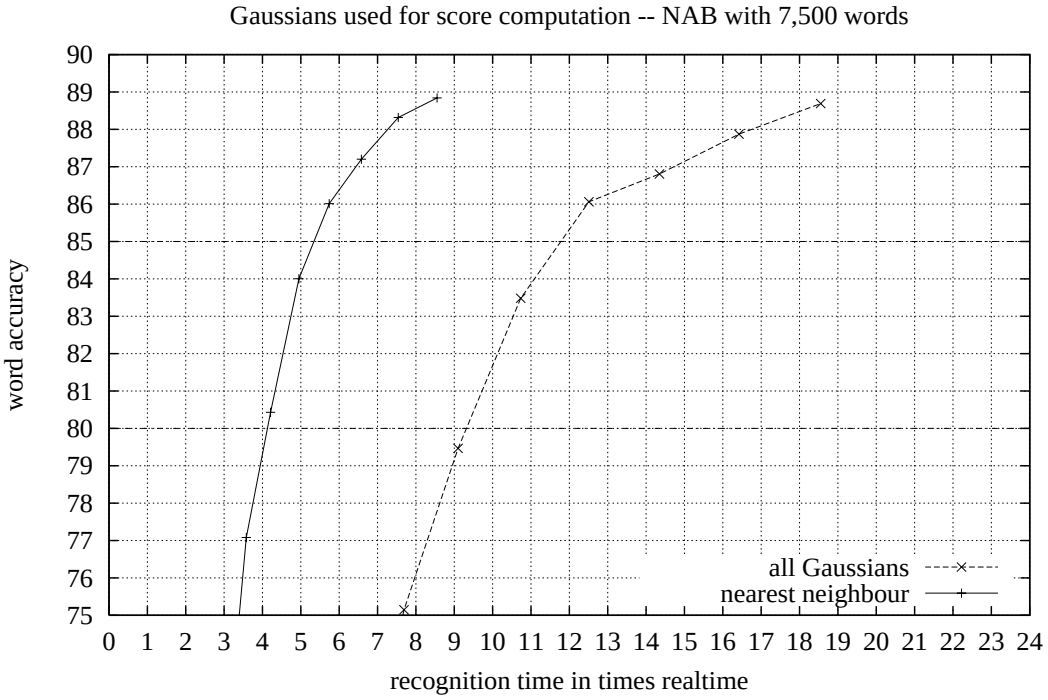


Figure 4.8: Word accuracy over recognition time using the nearest neighbor approximation. These experiments were performed on 50 sentences of female speakers, taken from th 1994 NAB evaluation data.

For the female subset of the NAB evaluation, no loss in word accuracy was observed. For the male subset, the loss was a mere 0.3% absolute word accuracy. On the *German Spontaneous Scheduling Task (GSST)*, the word accuracy dropped from 84.7% to 84.4%. Considering that this loss in word accuracy comes with a factor two to three boost of the recognition speed, this is nearly negligible.

Unless noted otherwise, all experiments in this thesis have been computed using the nearest neighbor approximation.

4.3.2 Gaussian Selection and related approaches

The goal of the *Gaussian selection* [Bocchieri, 1993b] is to reduce the number of Gaussians used for the computation of the observation probability.

As a first step, all Gaussians are clustered to build a smaller codebook, the GS-codebook. A GS-vector is associated with all Gaussians of the original codebook

that have a weighted distance below a threshold T to that GS-vector. Depending on this threshold T, a Gaussian of the original codebook can be associated with more than one GS-vector.

The computation of an observation probability is then performed in two steps: first, the GS-vector with the smallest distance to the input vector is determined; next, all Gaussians associated with that GS-vector are used for the main probability computation. Since there will be codebooks that have no Gaussian associated with some of the GS-vectors, the score for these cases is approximated by a constant back-off value.

As extension to this algorithm [Knill et al., 1996] proposes a two level algorithm to

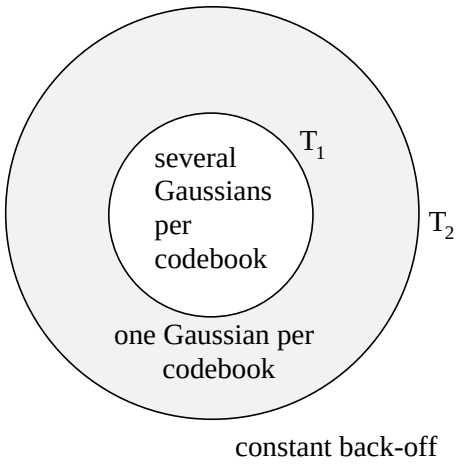


Figure 4.9: The double ring algorithm to approximate back-off values uses two thresholds, T_1 and T_2.

estimate the observation probability for Gaussians (figure 4.9): all Gaussians within a codebook that have a weighted distance smaller than T_1 to the selected GS-vector are directly used to compute the observation probability. If for a codebook there is no Gaussian in this area, the one Gaussian with the smallest distance to the GS-Vector is taken instead, provided that this distance is smaller than T_2. If the distance is larger than T_2 for all Gaussians of the codebook, the score is approximated by a fixed value.

One of the drawbacks of the Gaussian Selection algorithm is that if the number of GS-vectors is too small, the approximation is not good enough. If however the GS-codebook is large, there is a large overhead for finding the nearest neighbor in the GS-codebook.

In chapter 5.2.6 an alternative approach that is also based on segmenting the feature space is described: the Generalized-BBI algorithm, that was developed as part of this thesis, uses a binary tree instead of a second level codebook to segment the feature space. → 87

More algorithms for the fast computation of observation probabilities can be found in [Schukat-Talamazzini et al., 1993] and [Kurimo and Somervuo, 1996].

4.3.3 Neural Nets

→ 118 With the exception of the Abbot recognition system [Hochberg et al., 1995b], there are only few successful attempts [Fritsch et al., 1997] to estimate the observation probability for LVCSR systems using neural nets.

The Abbot system uses recurrent neural nets. During the training, multiple nets are trained for speaker clusters, which are merged prior to the recognition step.

The error rate of this system in the 1994 NAB evaluation on the P0 set was 12.4%. While the best completely HMM-based system had an error rate of only 7.2%, most of the remaining systems in the competition had only slightly lower error rates.

The big advantage of the Abbot system over systems that use mixtures of Gaussians to estimate the observation probabilities is, that the number of parameters required to estimate these probabilities with neural nets is smaller. Reasonable error rates can be acchieved even with context independent models or a very small number of context-dependent models (→ also [Tebelskis, 1995]). Therefore, the computation of the observation probabilities takes less time. Also, since the neural nets use a discriminative rather than a maximum likelihood approach, their discriminative ability is usually larger. That could reduce the risk of pruning the correct hypothesis. The Abbot evaluation system was reported to run at 15 RT on an HP735 workstation. This was about a factor of 10 faster than the competing systems.

Therefore, if using mixture of Gaussians is not a requirement, neural nets provide good alternative to estimate the observation probabilities.

4.4 Parallel Computing and Hardware

When running out of faster algorithms, it can be worth considering resorting to faster computers.

Multi-processor architectures with two to ten multi-purpose processors are getting more and more common. For these architectures, threads are an efficient way to parallelize the recognition process.

Some researchers and developers also resort to massively parallel architectures or special purpose hardware. However, many of these architectures are too restrictive for research purposes, and to expensive for most commercial applications. → 50

Finally, even among single processor machines there can be substantial differences in performance. Common benchmarks such as the SPECfp95 values can give an indication of how much faster a new platform is, but the actual performance for speech recognition can only be determined by running timings for the respective recognition system. → 51

4.4.1 Threads

With the DEC-Alpha 2100 servers, multi-processor machines have moved into many research groups that previously were not using parallel machines on a regular base. This trend has continued with the introduction of the SUN-Ultra-2 workstation and the SUN Enterprise servers. These machines have between 2 and 14 processors, which are often used to recognize one utterance per processor.

Threads provide a simple mechanism to parallelize the recognition with a much finer granularity.

For example, the computation of the observation probabilities for different codebooks can be executed in parallel on different processors. This is usually the most expensive subtask of the recognition process.

There is a distinct overhead for keeping a threaded process and to distribute the codebooks over the processors. Table 4.1 shows the results for parallelizing the score computation with threads.

	Real-time
without Threads	0.80 * RT
with Threads	0.74 * RT

Table 4.1: Example for a NAB system using a vocabulary size of 7,500 words at 83.2% word accuracy on a SUN-Ultra with two CPUs.

As the computation time for the optimized NAB system with *Lookaheads* is spread over many routines, an efficient parallelization within one utterance becomes difficult. If the workstation has enough memory to process two utterances at a time,

and if the application does not require a fast serial processing of single utterances, it is more efficient to parallelize over several utterances.

4.4.2 Special Purpose Hardware

Product oriented Hardware

One application for special purpose hardware is to market products that cannot run in real time on common PC platforms. One example for this approach is the Siemens Dictation System [Niemöller et al., 1997] that uses a PC-board with several special purpose DSP chips to compute the observation probabilities.

The advantage of this approach for products is that it is much more difficult to make illegal copies of custom hardware compared to speech recognition software. If however the competition is able to offer an all-software product, the hardware solution may become hard to sell.

Research oriented Hardware

There are many parallel architectures that could theoretically improve the research possibilities for the development of speech recognition algorithms, such as iWarp, MASPAR [Sloboda, 1992, Jürgens, 1995], CNAPS [Wahle, 1994]. However, very few research groups are using them on a regular basis.

The main problem with these approaches are:

- Fix-point arithmetic, lack of memory and other restrictions can severely limit the application of parallel computers for many speech algorithms.
- If algorithms that have been optimized to sequential architectures are ported to massively parallel platforms, some of the optimizations have to be reversed. Also, not all parts of the program can be efficiently parallelized. The actual speed-up of the total program is always substantially smaller than the theoretically possible speed-up.
- Research prototypes have to be flexible and change quickly to accommodate novel research ideas. If the effort of porting existing software to the special purpose hardware is high, new approaches will not be ported and the parallel version quickly becomes outdated.
- Some of the massively parallel computers are too expensive for their useful lifespan. Even if a machine is 50 times faster than a conventional workstation on a real world problem, this advantage can quickly be out-paced by the development of new sequential processors.

4.4.3 Faster Processors

To evaluate speech recognition systems, the word accuracy or word error rate is used. However, it is often more interesting to know how well a given system or algorithm performs at a given speed.

Measuring the word accuracy to compare algorithms implemented on different recognition systems can already be challenging: the influence of other differences between the systems, including but not limited to the number of bugs in the code and training procedure, the amount and quality of training data, and the ability of the researcher to make good educated guesses for the initial setting of the many parameters of such a system, render most of these types of comparisons futile.

If at the same time the speed of the recognizer needs to be taken into account, matters get worse: most systems have not been developed to run on the same hardware using the same compiler. Even if a system is ported to a new platform for comparison with another system, the system that has been developed on that platform has an advantage because it has likely been tuned to the strong points of the original architecture.

Many publications do not mention what configuration was used for the published results. If this information is available, careful estimates can be made using standardized processor and machine benchmarks such as SPECint92/95 and SPECfp92/95 for integer and floating point performance respectively. However, care should be taken to make sure these numbers can be applied to speech recognition.

Most timings published in this thesis have been performed on a single processor of a two processor SUN-Ultra-2 workstation operating at a clock rate of 167 MHz.

The following table give a list of SPEC values for some machines commonly used in the speech community. These values were taken from the world wide web[1].

Machine	CPUs	MHz	Fp92	Fp95
Sun Ultra 2	1	167	351	9.33
Dec Alpha 2100-4/275	1	275	291	-.–
HP 9000 Series 700 Model 735	1	99	174	3.98
SGI Indigo2 R4400SC	1	150	94	-.–

Table 4.2: SPEC values for some common machines

Experiments

To evaluate the predictive power of the SPEC-results for a given speech recognition system, the author has performed comparative timings for one experiment over a small selection of platforms.

[1] `http://www.specbench.org` Januar 1998.

Since compiler options and minor rearrangements of the code can substantially influence the results in some cases, these experiments, like all comparative timings between platforms, should be considered with some caution.

The test conditions for the results in table 4.3 were as follows:

Compiler and compiler options

- **gcc:** For all machines gcc version 2.7.2 was used.
  `gcc -O3 -fomit-frame-pointer -ffast-math`
- **cc on Sun:** Version SC4.0 (18 Oct 1995) was used. The compiler options reflect quite a bit of tuning, though not exactly to this but to similar recognition tasks:
  `cc -xO5 -fsimple=1 -dalign -native -Xa`
- **cc on DEC-Alpha:** The native compiler delivered with OSF1 V4.0 was used. The compiler options reflect the same amount of tuning as those for the Sun compiler:
  `cc -O4 -Olimit 2000 -float_const -fp_reorder -no_misalign -tune host`
- **cc on HP:** The native compiler delivered for HP-UX B.10.01 A was used. All attempts at more aggressive optimization lead to a linker error.
  `cc -Ae +O4`

Machine	Clock	Compiler	Real-time	Memory
Sun Ultra-2	167 MHz	cc SC4.0	1.03	262 MBytes
Sun Ultra-2	167 MHz	gcc 2.7.2	1.13	262 MBytes
DEC Alpha 2100-4/275	275 MHz	cc	1.80	394 MBytes
DEC Alpha 2100-4/275	275 MHz	gcc 2.7.2	1.82	394 MBytes
HP 9000/735	99 MHz	cc	1.77	228 MBytes
HP 9000/735	99 MHz	gcc 2.7.2	1.90	228 MBytes

Table 4.3: Comparative timings for 7,500 Word NAB for a word error rate of 13.4%.

Test environment

Using janus3.2-P000, a test on the 50 female NAB sentences was performed. The test was using the Generalized-BBI algorithm, lookaheads, and frame-skipping for best performance. The vocabulary was limited to 7,500 words to make sure the program would fit completely into the main memory of all machines used without swapping.

The test was repeated four times in the same process to avoid distortions due to swapping during the process startup. The presented times reflect the user time without startup as reported by the c routine `times` and verified using the `/usr/bin/time` command. The largest and smallest value of the four runs were removed before computing the average over the remaining two runs. The deviations between any two runs on the same machine were within a 3% margin.

Conclusion

The SPEC92fp benchmarks give little indication on how well JRTk will perform on a machine. While the comparison between the HP machines and the Sun-Ultra turns out just as expected from the SPEC92fp values, the timings for the Alpha processor are surprisingly slow. One reason for this could be that the code is optimized toward 32-bit operations while the Alpha architecture works most efficiently for 64-bit operations. Note also, that these experiments were performed for a recognition system that relies heavily on float operations and may not port at all to systems that have been optimized for integer performance. The only reliable way to tell how fast a given speech recognition system will run on a new platform, therefore, is to try an actual recognition run.

Chapter 5

Thesis Contributions

We are all agreed that your theory is crazy. The question which divides us is whether it is crazy enough to have a chance of being correct. My own feeling is that it is not crazy enough.
– Niels Bohr

In this chapter three different approaches to reduce the recognition effort are investigated. By combining the resulting algorithms, real-time performance for a 65,000 word NAB recognizer was achieved starting from an NAB evaluation system that ran about 200 times slower.

First, search issues like the implementation of the tree search, pruning strategies, and lookaheads will be addressed. The main idea behind the algorithms presented in this first section is to reduce the recognition time by avoiding expensive computations for part of the search dictionary. Some of the techniques described here, such as the tree search, were already used to make the 200 RT evaluation system possible in the first place. → 56

The second section is about reducing the effort for the most expensive subtask in the recognizer, namely the computation of the observation probabilities with mixtures of Gaussians. Balancing the quality of the computed scores against the quantity of scores that can be computed per second is an important issue in this section. The most important algorithm presented in this section is the *Big-Bucket-Box-Intersection* algorithm, that was found to be best suited to reduce the computation effort while maintaining a high accuracy. → 76

Finally, algorithms designed to avoid expensive computations that are not likely to produce new information in stationary sections of the speech signal conclude the chapter. → 96

Since the three directions are sufficiently orthogonal to each other, the individual gains add up to the total required speed-up. Each section builds on the previous sections and shows the possible speed-ups based on the best system so far.

5.1 Search

The multi-pass search algorithm used for this thesis is designed to work best for the recognition of unrestricted continuous speech using large vocabularies between 2,000 and 65,000 words.

To capture coarticulation effects between words, the search has to allow for acoustic models that depend on the previous or following word. The resulting problems and solutions are briefly described in the first part of this section, as they have an important effect on the complexity of the recognition task.

The search itself is organized into two time synchronous forward passes. The output of each pass is a *backtrace*, that can be used to restrict further passes, to build word graphs, or to simply extract the best hypothesis. The first pass is a tree search using an approximation to include trigram probabilities that is very similar to the
→ 37 *Delayed Bigrams* described in the previous chapter. The second pass is a linear search that is primarily used to recover errors made because of the approximations in the tree pass, but can also use more complex acoustic models.

After introducing the two search passes, the pruning mechanisms used in the first search pass are presented along with some experiments on their respective benefits. The last two parts of this section describe the language model and acoustic lookaheads.

5.1.1 Cross-Word Models

If the allophone to be used for a phoneme depends on the successor word, the actual model cannot be determined before the identity of the successor word is known.

Unfortunately, before recognizing the successor word the current word has to be recognized. One way out of this dilemma is to create copies of the current word for each possible successor word, then recognize the successor word and only keep the copies that were actually required. However, to identify the correct successor word, it may theoretically be required to know the next word too.

Fortunately, it is usually safe to assume that the cross-word coarticulation is limited to phonemes that are close to the word boundaries.

In JRTk, only the models for the first and last phoneme of a word can depend on cross word contexts. Also, this context is limited to the last phoneme of the predecessor word and the first phoneme of the successor word respectively.

To avoid a combinatorial explosion, different techniques are used to manage left contexts, right contexts, and single phone words.

Left Context

For cross-word dependent models, the allophones to be used for the first phoneme in a word depend on the last phoneme of the predecessor word.

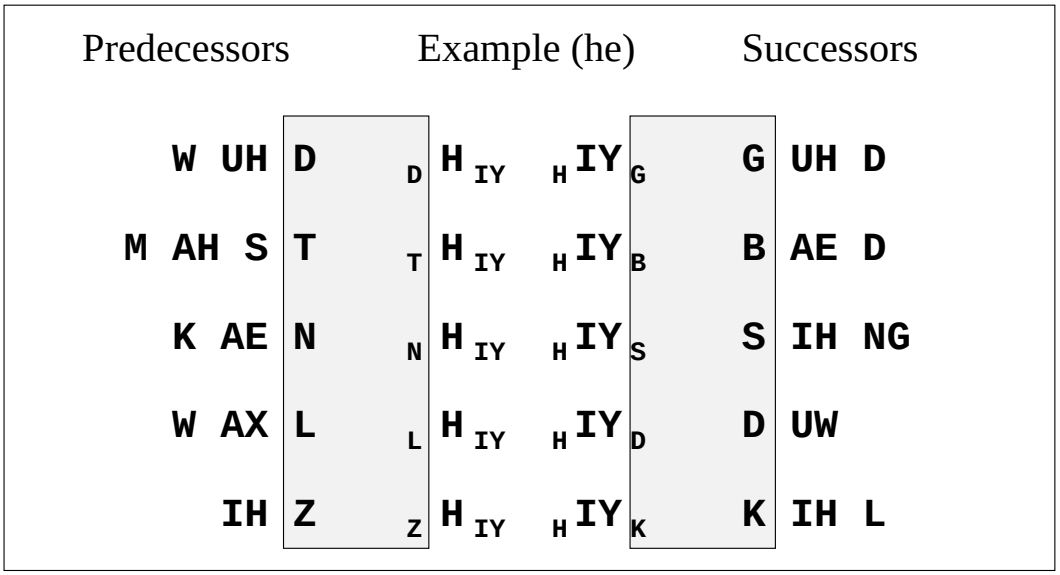


Figure 5.1: context dependent modeling of the word *he*, only shown with 5 of the 50 possible context phonemes to each side. The words used in this example are *would, must, can, will, is* and *good, bad, sing, do, kill*.

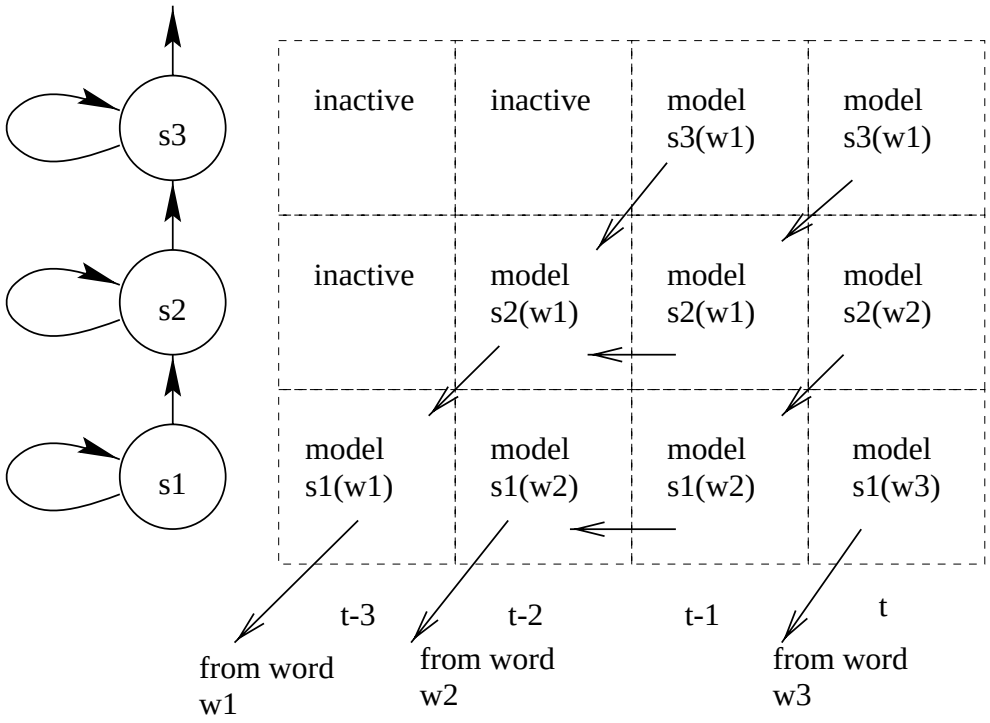


Figure 5.2: Example for a multiplexed word-begin phoneme.

To determine the left context, the identity of the predecessor word is required. The left context is modeled using the identity of the last phoneme of the best predecessor word ending at this frame. For search passes that are using *Delayed Bigrams*, this predecessor has to be picked without using any language model information.

The identity of the best predecessor word can differ from frame to frame. Since the partial hypotheses for the different states of a word-begin phoneme did not made the transition into this word at the same frame, the states within one phoneme can have different predecessor words and use different allophone models. Also, the allophone model to be used for one state can change from frame to frame. In JRTk those phonemes are multiplexed as follows: each state is modeled using the context that corresponds to its best predecessor word. The sub-allophone used to model a state in a word-begin phoneme therefore depends on previous decisions in the search.

Figure 5.2 shows how the multiplexed phoneme models work. In the example there are three active word ends w_1, w_2 and w_3 that have a similar cumulative score. The sub-allophone used to model the states of the word-begin phoneme depends on the predecessor word for this state. However, this solution is not exact: at the time t the word w_2 is not available as predecessor for state s_3, since all legal predecessor states use w_1 as their best predecessor. If the best overall chain was indeed $s_1(w_2, t-2), s_2(w_2, t-1), s_3(w_2, t)$ it would be lost due to earlier decisions.

Right Context

Because most words can be pruned before the search reaches their last phoneme, it is possible to make dedicated copies of the remaining word-end phonemes for all possible right contexts. However, if the pruning thresholds are set to allow too many word ends, and if the number of possible right contexts is large, the computational effort for this approach can quickly dominate the time required for the recognition process.

Single Phone Words

Which allophone to use for single-phone words depends on the last phoneme of the predecessor word and on the first phoneme of the successor word. To avoid the overhead for evaluating up to 2500 possible combinations, single-phone word contexts are restricted to the predecessor word and use a multiplexed phoneme model.

5.1.2 Tree Search

→ 41

For the tree search, the lexicon is organized as an allophone-tree, or more precisely as a forest of allophone-trees. All phonemes that model the beginning of words using the same allophone share the root of one of these trees. Trees built for context dependent systems are much less compact than trees for context independent systems. The context dependent NAB recognizer currently in use has over 1,000 root nodes, depending on the vocabulary size and the complexity of the acoustic modeling, while context independent systems need only about 50 root nodes.

The nodes further down the trees represent the allophones for the middle sections of the words. The leaves of the tree correspond to the word-end phonemes.

At the transition from the word ends (the leaves of the trees) into a new word (a root of a tree), the identity of the new word is not known. The transition is therefore computed using the leaf with the best cumulative score so far as predecessor. The only language model included at this point is the unigram lookahead. → 69

Further down the tree, upon computing the transition into the last phoneme of the word, the word identity is always known. At this point the best predecessor for the current word given the trigram language model and the fixed entry point for this word is determined.

As an additional approximation, it is assumed that the best predecessor word of the predecessor word, w_{-2}, is correct and independent of the current word. Then, the best predecessor word for w_0 is the w_{-1} that minimizes the sum

$$\text{word-end-score}(w_{-1}(t_{beg-1})) - log(P(w_0|w_{-1}, w_{-2}))$$

This predecessor word is used to compute the language model probability that replaces the unigram lookahead at the transition into the last phoneme. This *Delayed Trigram* approach is very similar to the *Delayed Bigrams* described in chapter 4.1.2. → 37 However, it allows the use of some trigram information in the first search pass.

5.1.3 Linear Search

The linear search pas uses a normal, linear search lexicon. The search space is restricted using information obtained from the backtrace of the previous search pass, usually a tree search. → 28

Based on the backtrace, a word list is built for each frame, containing all words that will be allowed to start at that frame in the second pass. Each word that reached its word end for longer than a given amount of frames in the first pass is considered a good candidate and is allowed to start at the corresponding entry frame in the second pass. Since it is known that the tree pass has introduced segmentation errors into the search, the area where a word is allowed to start is expanded by a few frames in either direction. → 37

Figure 5.3 shows an example where the word has to reach the word end for a minimum of two frames to be considered as candidate for the second pass. The startup region is expanded by one frame in each direction[1].

These word lists are used to limit the number of words that are considered for word transitions in the linear search. Thus, the vocabulary at each frame is reduced enough to make a linear search viable.

[1] More realistic values are 5 frames for the number of frames a word has to reach its word end and a 15 frame error margin for the width of the start window.

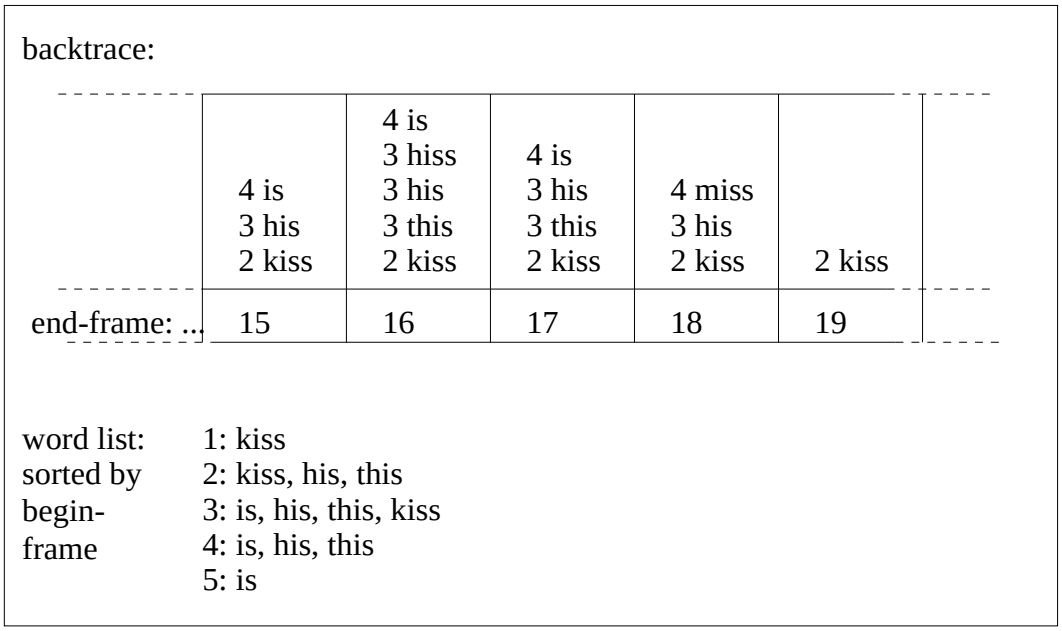


Figure 5.3: A segment taken from a backtrace. To build the word list, the words from the backtrace are examined and added to the word list of the corresponding entry frame (shown to the left of the words). The words '*hiss*' and '*miss*' did not make it into the word list because they only reached their word end for a single frame.

The linear search includes the language model information directly at the transition into the new word, avoiding the segmentation errors the tree search may have made. However, the best predecessor is still chosen assuming that the predecessor of the predecessor depends only on the identity of the predecessor and not on the current word (*poor man's trigrams*). This approximation is dropped for the language model re-scoring of the word graphs.

Experiments

In these experiments, the linear search is used to repair the 12% relative error rate increase introduced by using *Delayed Trigrams* instead of full trigrams in the tree search. At high word accuracies, the maximum yield in word accuracy for the linear pass therefore is 1.2% absolute. However, to catch all these errors, the required computation time for the linear search pass alone is about 40% of the time required for the tree search. As shown in figure 5.4, the linear search should therefore only be used in areas where the plot of the word accuracy over the runtime is already rather flat.

In the case of the NAB system with 65,000 words, the linear search pays off only for word accuracies over 87% and recognition times beyond 2.5 times real time. For faster systems, a larger increase in word accuracy can always be obtained by relaxing the pruning thresholds.

The combined search can also use a simpler acoustic model in the tree than in the

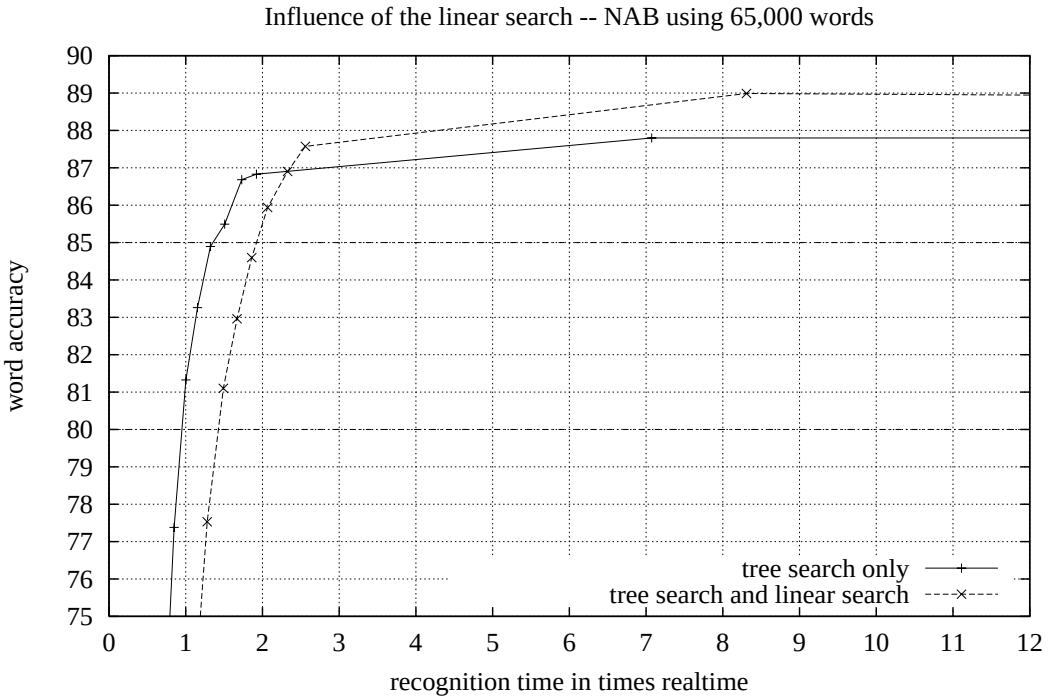


Figure 5.4: tree and linear search versus tree search only.

linear pass. That way, the recognition time in the tree search goes down, and the difference in word accuracy between the tree and linear passes increases. However, if the tree uses a less accurate acoustic model, it is less effective in reducing the search space for the linear pass. Preliminary experiments on such techniques did not yield any improvements over using just the tree search for fast systems.

5.1.4 Pruning the Search Space

The default pruning paradigm in JRTk is a beam search, where the cumulative score of the partial hypotheses is used to remove all partial hypotheses from the search space that differ by more than a threshold from the best partial hypothesis at the time. These thresholds are also called *beams*. → 39

An alternative approach is to restrict the search space to a maximum number of active partial hypotheses, phonemes, or word ends. These pruning strategies are especially useful to provide an upper bound to the total search effort.

Beams

In this thesis, up to five different beams for the tree search, and four more for the linear search were used. The main reason for the large number of beams is to provide a platform to investigate whether different pruning thresholds for different points in the search are useful. → 63

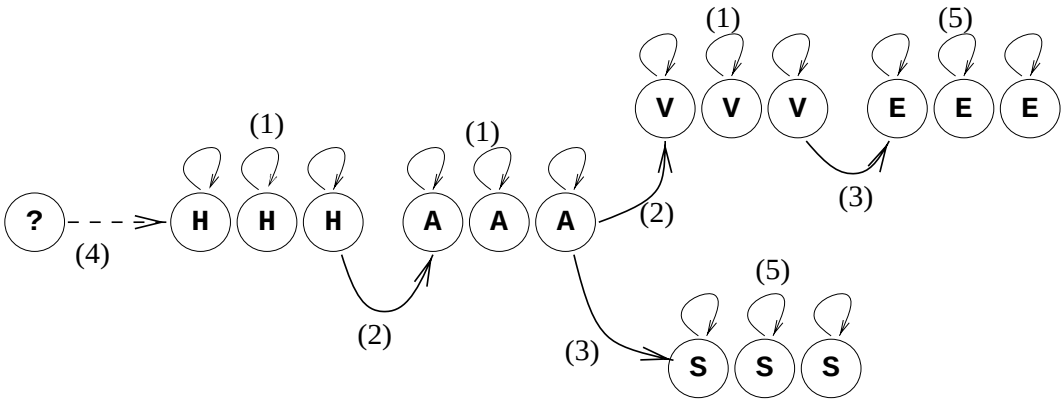


Figure 5.5: Use of different beams in JRTk. The numbers in this figure correspond to the description in the text.

1. **Pruning of roots and nodes:**
   If the best cumulative score of all partial hypotheses *within a root or node* of the search tree exceeds the cumulative score of the best partial hypothesis at the time by more than this beam, all states within this phoneme are set inactive.

2. **Transitions into nodes:**
   If the cumulative score for the partial hypothesis in the *last state of a phoneme* exceeds the cumulative score of the best partial hypothesis at the time by more than this beam, no transition from this state into the next phoneme is considered.

3. **Transitions into leaves:**
   If the cumulative score for the partial hypothesis in the last state of a model *that is succeeded by a leaf* exceeds the cumulative score of the best partial hypothesis by more than this beam, the transition into this leaf is not computed. Otherwise, the transition is computed and if this transition activates a formerly inactive leaf, all cross-word dependent copies of this leaf are allocated.

4. **Word to Word Transitions:**
   If the cumulative score for the partial hypothesis in *the last state of a leaf* exceeds the best cumulative score in all leaves by more than this beam, no transitions (into roots) are computed from this state.

5. **Pruning leaves in tree search:**
   If the best cumulative score of all states *within one leaf* exceeds the best cumulative score in all leaves by more than this value, all states in this leaf are deactivated. The reason why leaves are only compared to leaves in the tree search that the language model penalty is added at the transition into the leaf. Therefore, it is hard to compare the score in this model to the scores in the states right before the transition. Since there are no such *Delayed Bigrams* in the linear search, this beam is only used for the tree search.

The large variety of pruning beams may appear confusing. However, for most cases their individual settings have only little influence on the total system performance and a novice user can safely set them all to the same value. → 65

Maximum Number of Active Word Ends

To use a limit for the number of active word ends is mostly a security feature: even if all other beams are set to extremely large values, the maximum number of word ends remains reasonable. For small beams, the number of actually active word ends is still allowed to drop below this value. This pruning strategy avoids unwanted growth of the search space for inputs that do not match the recognizers models well, and limits the resulting recognition time.

Since the memory required to store the backtrace depends directly on the number of word ends, limiting the maximum number of word ends also controls the maximum amount of memory used per frame.

Even for a vocabulary size of 65,000 words, the maximum number of word ends can be reduced to as little as 15 to 50 words without a significant loss in word accuracy.

The biggest advantage of using the number of word ends rather than an absolute score threshold is that this number does not depend on the size of the scores of the acoustic modeling used. If a new acoustic model is to be used (e.g. with more coefficients per feature vector), these values do not need to be readjusted.

Using a constant rather than a maximum number of active word ends is not useful to restrict the search space, since the number of required word ends varies within a sentence and from sentence to sentence. Usually, the normal beams model these changing requirements better than a constant number of hypotheses.

Maximum Number of active Phonemes

Similar to the maximum number of word ends, this measure limits the maximum number of active phoneme models. Whenever a constant value of active phoneme models is exceeded, all active phoneme models are sorted by the best cumulative score in the model. All but a fixed number of models are then deactivated, leaving only the best ones active. Again, this value is mostly independent of the actual modeling used, and can serve as an emergency limit to avoid excessive search time.

Experiments

This chapter studies the effect of the settings of the experimental beams provided by the JRTk search. → 39

Four sets of experiments were performed:

1. All pruning thresholds get the same setting.

2. All pruning thresholds get the same setting, and the maximum number of active word ends is restricted.

3. The relative setting of the five thresholds in the tree search is adjusted using values derived from *a priori* knowledge.

4. The relative setting of the same five thresholds is hand tuned on the test set to estimate an upper limit for the gain using several beams.

One threshold

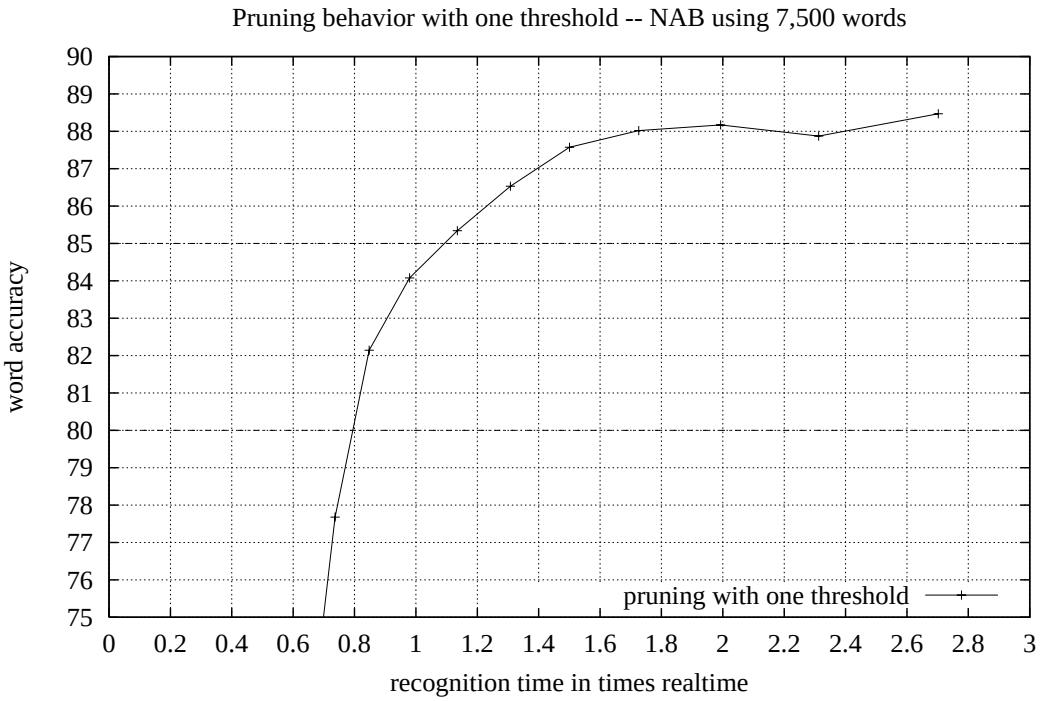


Figure 5.6: Pruning with one threshold. The RT=0.74 WA=77.7 value corresponds to a pruning threshold of 170. The RT=1.73 WA=88.0 value corresponds to a pruning threshold of 230.

For this experiment, all thresholds are set to the same value, while all other pruning mechanisms are disabled. A partial hypothesis that matches the current input considerably worse than the currently best partial hypothesis for several phonemes is not likely to lead to the globally best hypothesis. Therefore, the size of the single pruning threshold should roughly correspond to the cumulative score over one to two phonemes.

The average score per phoneme for the NAB system used for this experiment ranges around 130. Therefore, useful values for pruning beams would be expected to lie

between 130 and 260. Figure 5.6 illustrates how the word accuracy changes with the recognition speed for different values of the single pruning threshold.

Maximum number of active word ends

The required value for the maximum number of active word ends can be found on a cross-validation set as follows: For all hypotheses produced for the cross-validation set, the ranks of all word ends in the *backtraces* are determined. If only word ends that have a higher rank are removed, the original hypotheses can still be found. But for very few sentences, a value of 15 to 20 active word ends is sufficiently large.

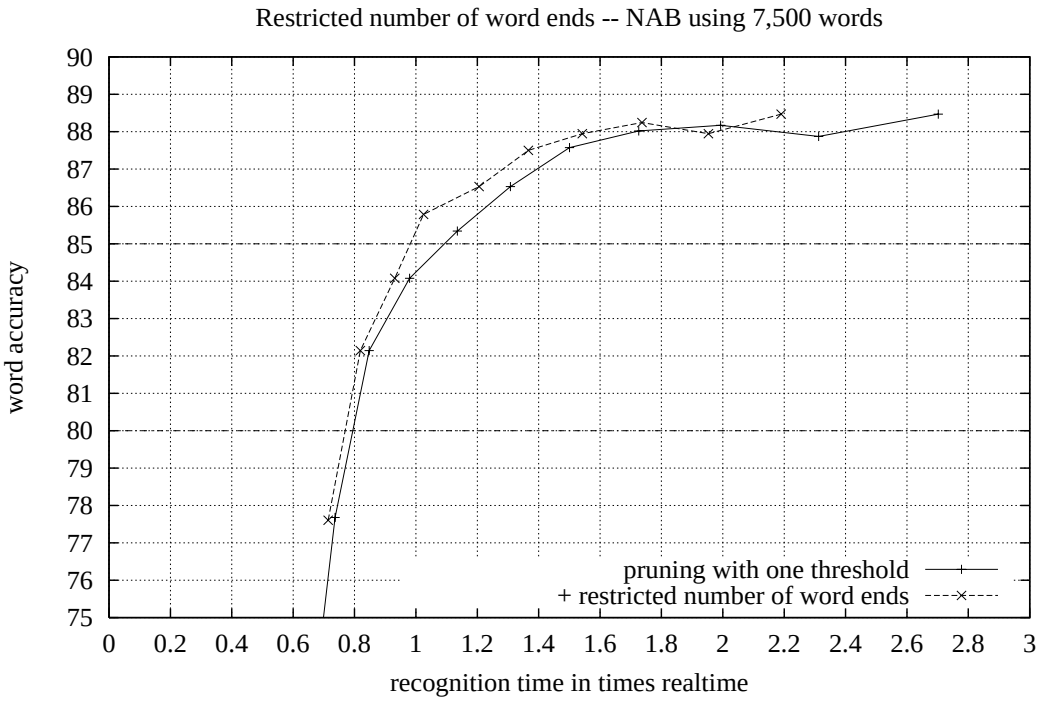


Figure 5.7: Pruning with one threshold and a limited number of active word ends.

Especially in systems that use conservative settings for the remaining beams, reducing the maximum number of active word ends yields considerable speed-ups. Figure 5.8 illustrates how using the number of active word ends to restrict the search space puts an upper bound to the recognition time. This makes it easier to guarantee that the response time of the system remains within the limits required for a given application.

Five pruning thresholds

The following considerations went into the initial adjustment of the five pruning thresholds relative to each other:

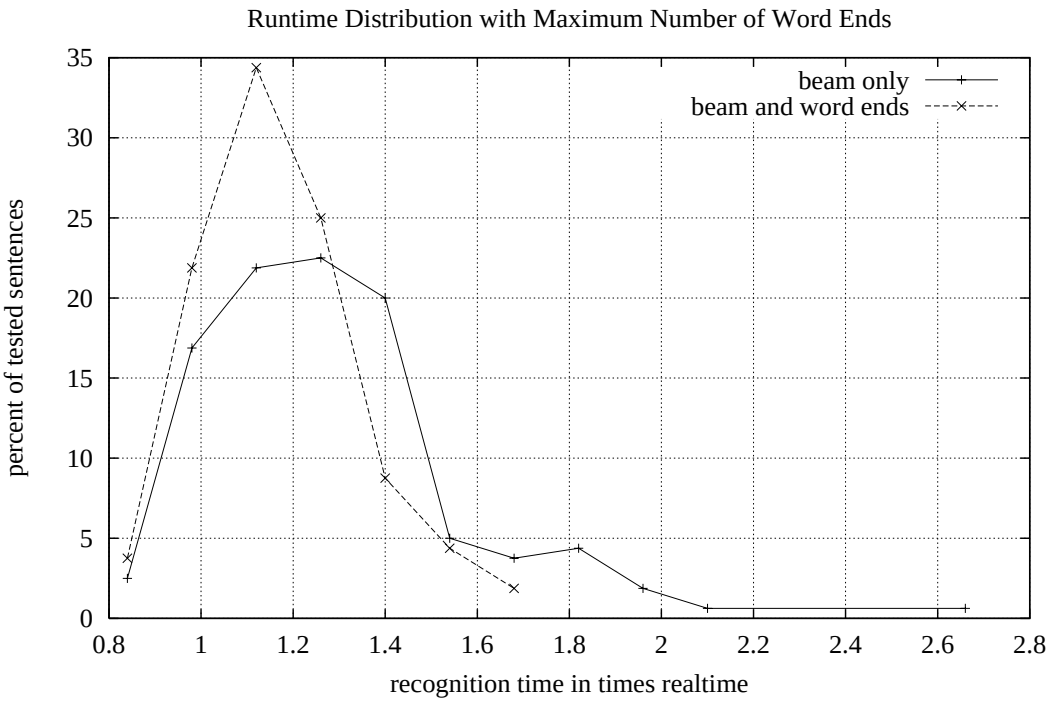


Figure 5.8: Runtime distribution using a limited number of active word ends.

1. The more specialized a threshold is, the smaller it should be.

2. If the system commits a search error due to too small beams, the resulting gain in computing time should be as large as possible. The more expensive a step in the search is (cross-word polyphones with phoneme copies at word ends), the more aggressively should the pruning try to minimize the number of active states for this step.

Figure 5.9 shows that the word accuracy for a search that is pruning with five thresholds is a little higher than for a search that is using a single threshold. These five initial *a priori* pruning beams have been used for all NAB experiments presented in other sections of this thesis to plot the word accuracy over the recognition time.

Manually tuned thresholds

To estimate an upper limit for the possible improvement of five beams versus one, the relative setting of the beams was manually tuned on the test set. To that end, the following procedure was carried out for each beam, beginning with the within phoneme beam: the beam is set to a number of different values, and for each of these values the other beams are varied according to the settings in table 5.1. Whenever better settings are found, the table is updated for the remaining experiments. Figure 5.10 shows the difference between the *a priori* and tuned versions of the beams.

	top	b	pb	lpb	wb	lpab
faster	..	...	...	...	...	...
..	16	160	150	140	140	135
..	17	170	160	150	150	145
..	18	180	170	160	160	155
..	19	190	180	170	170	165
..	20	200	190	180	180	175
..	21	210	200	190	190	185
..	22	220	210	200	200	195
..	23	230	220	210	210	205
..	24	240	230	220	220	215
..	25	250	240	230	230	225
better	..	...	...	...	...	...

Table 5.1: initial pruning thresholds; **top:** Maximum number of active word ends; **b:** Pruning within phonemes; **pb:** Pruning at phoneme transitions; **lpb:** Transition into word-end phonemes; **wb:** Word transitions; **lpab:** Word-end phonemes among themselves.

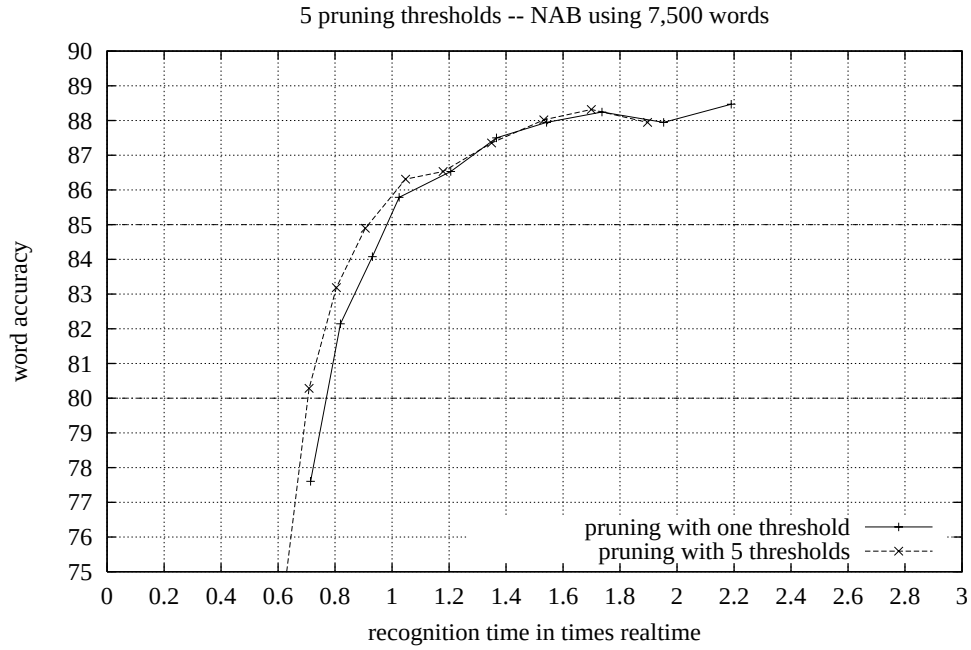


Figure 5.9: pruning using five *a priori* beams.

Number of active word ends: The word accuracy drops at settings of less than 10 active word ends. Larger settings lead to a higher recognition time for large beams only. Limiting the number of active word ends also reduces the variance of the recognition time for different sentences.

Pruning of the last phonemes among themselves: The original values were a little larger than required.

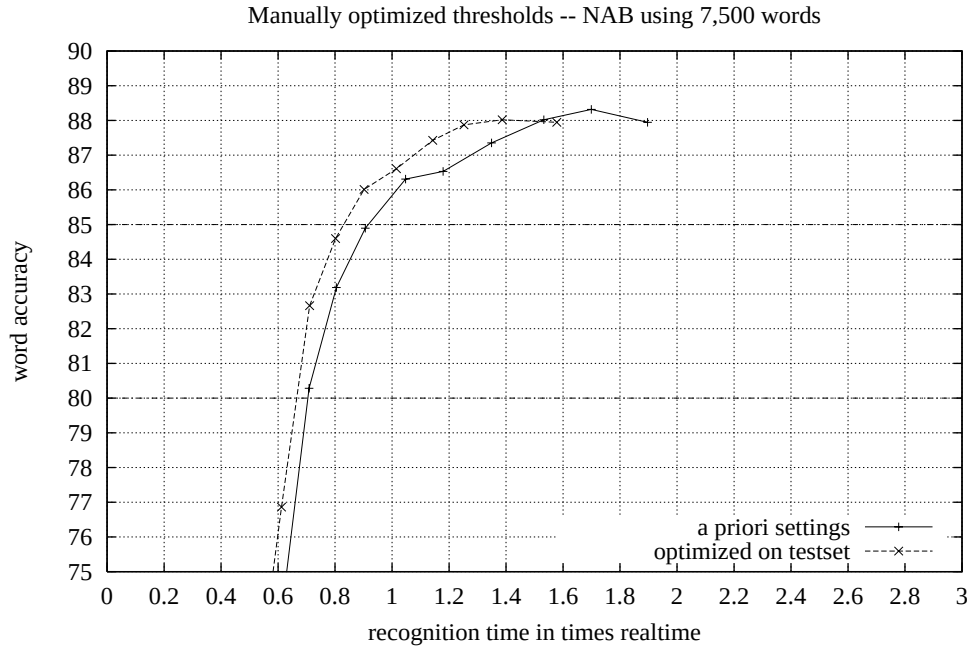


Figure 5.10: Beams tuned on the test set versus *a priori* beams.

Other thresholds: All improvements seen in figure 5.10 were due to the improved settings of the maximum number of active word ends and the pruning of the last phonemes among themselves. For the remaining beams the *a priori* values were close to optimal.

Pruning thresholds: Summary

Fine tuning the relative settings of the pruning thresholds pays off in areas where the error rates are small and the pruning allows large gains in recognition time for small losses in word accuracy. However, the yield of using five instead of one pruning threshold is small: even at a high word accuracy of 88%, the maximum speed-up for the 7,500 word NAB system is only around 12%. Fine tuning the thresholds should therefore be the finishing touch to a system.

5.1.5 Unigram Lookaheads

When using a tree structured lexicon, the bigram information cannot be included until after the word identity is known. → 41

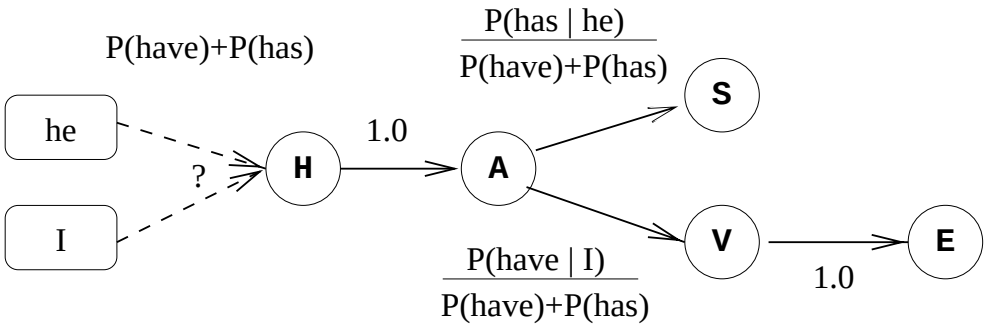


Figure 5.11: An example for language model *lookaheads* for the transitions in a tree with the words *have* and *has*. Until the bigram probability can be included, an approximation of the unigram probability has to be used.

However, estimates for the unigram probability of the remaining subtree can be calculated before reaching the final phoneme. They can be used to remove partial hypotheses that combine a bad cumulative score with an unlikely list of possible word ends as outlook.

In figure 5.11 the unigram probability for the transition of going into *either* one of the two words in the tree can be included the moment the transition into the root of the tree is computed. The resulting combined score allows a more accurate comparison to partial hypotheses in other tree roots. At the transition from the phoneme `H` into the phoneme `A` the number of words in the remaining subtree is still the same, so no further information can be provided. However at the transition into the `S` of the word *has* the identity of the word is known. At this point the best predecessor for the corresponding entry point is determined and the unigram probability used so far is replaced by the bigram probability $P(has|he)$.

This lookahead strategy has been previously published in [Woszczyna and Finke, 1996, Steinbiss et al., 1994].

Experiments

Unlike the JRTk-driven experiments in the rest of this thesis, the results presented in this section have been produced using an older NAB recognition system based on the JANUS-2 code. The maximum word accuracy for the experiments in this section is therefore considerably less than in the remaining sections. The *Unigram-Lookaheads* evaluated here are now an integral part of the JRTk search engine.

Also, unlike in other sections of this thesis that always plot the word accuracy over the recognition time in times real time, the graphs in this section show the word accuracy over the number of computed observation probabilities used per frame.

For JANUS-2, the number of score computations was roughly proportional to the recognition time. It was chosen for these experiments because it is independent of the computer used to perform the experiments.

Data points within one line show the pruning behavior for different word-end thresholds, graphs with the same type of marker differ in the setting of the within-phoneme pruning threshold.

Figure 5.13 illustrates the loss in word accuracy due to segmentation errors that has
→ 37 already been described in chapter 4.1.2. Below a certain pruning threshold, these search errors introduced by the use of the *Delayed Bigrams* can no longer be repaired by the second search pass.

Using *Unigram Lookaheads*, an estimate for the language model penalty is already available at the beginning of the word, and this estimate is updated at each phoneme transition. This reduces the risk of pruning the globally best hypothesis. As can be seen in figure 5.12, the word accuracy for smaller beams is considerably higher with *unigram lookaheads* than without.

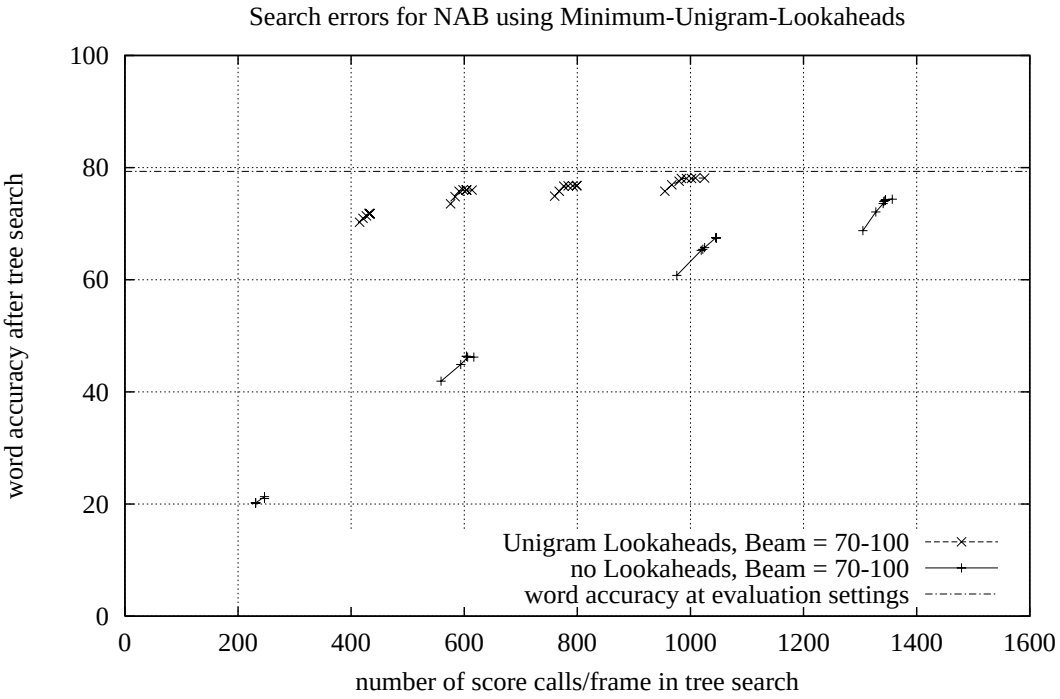


Figure 5.12: Reduction of search errors using *Unigram-Lookaheads* on NAB after the first search pass (tree search).

As an alternative approach, using the largest bigram probability instead of the largest monogram probability has been tried. For each phoneme transition in the tree, the largest bigram probability for all possible predecessor words and all words remaining in the subtree is computed and used as an estimate for the expected bigram probability. Though this procedure is considerably more complex than the

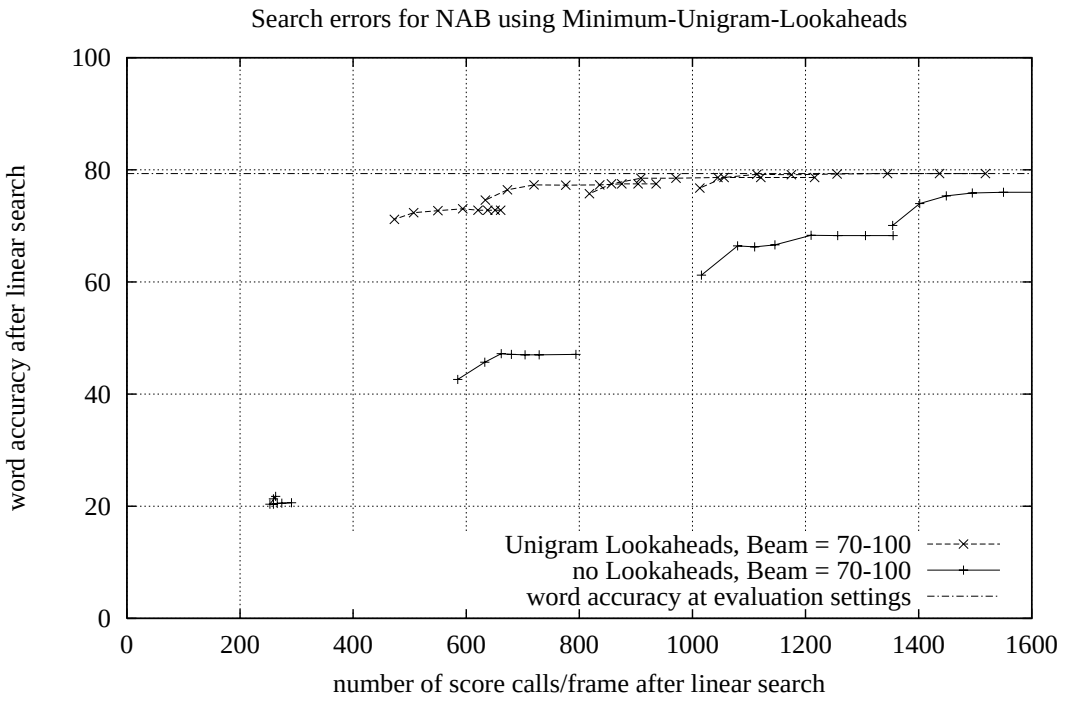


Figure 5.13: Reduction of search errors using *Unigram-Lookaheads* on NAB after both search passes (tree and linear search).

unigram lookahead, it failed to outperform the simpler approach and is therefore no longer used.

5.1.6 Phoneme Lookaheads

One of the most important techniques in reducing the recognition effort is the use of acoustic lookaheads at phoneme level. Their job is to predict at as little cost as possible, how well each partial hypothesis will do in a couple of frames. This estimated future score can be used to prune bad hypotheses before more costly operations are computed for them.

For the transition from the end of one phoneme into the first state of a new phoneme it is possible to predict how well this new phoneme will do over the next couple of frames by using a simpler acoustic model for that phoneme. By combining the score of the partial hypothesis prior to the transition and this lookahead score, a more accurate comparison of this partial hypothesis to other hypotheses can be done to decide whether or not this transition is worth trying.

Since even a tree search has many partial hypotheses that continue with the same phoneme or monophone, using an acoustic lookahead for more aggressive pruning can reduce the computational effort significantly. In context dependent systems the tree search can have over 1,000 root nodes which are hard to prune because in the first states of a root node very little acoustic and language model information is

available. Using acoustic lookaheads makes it possible to predict which of those root nodes will be pruned within the next couple of frames and so avoid to activate them in the first place. The effort of computing the scores for about 50 context independent lookaheads is small compared to the effort of computing the scores for the 1,000 context dependent root nodes.

In this thesis the usability of context independent *hidden Markov models* to estimate acoustice lookahead scores was investigated, because this approach fits well into the framework of an HMM-based recognition system. For most HMM-based recognizers a small context-independent system is built during the bootstrap process. This system can be reused for the lookaheads, eliminating the need of training additional models.

Another method of estimating lookahead scores is to use the output of a context-independent neural net that takes a couple of frames as input and is trained to predict which monophones are likely to start at a given point [Paschen, 1997].

Other information on phoneme-lookaheads and fast matches can be found in [Kenny et al., 1993], [Garudadri et al., 1994], and [Ravishankar, 1996].

HMM-based Phoneme-Lookaheads

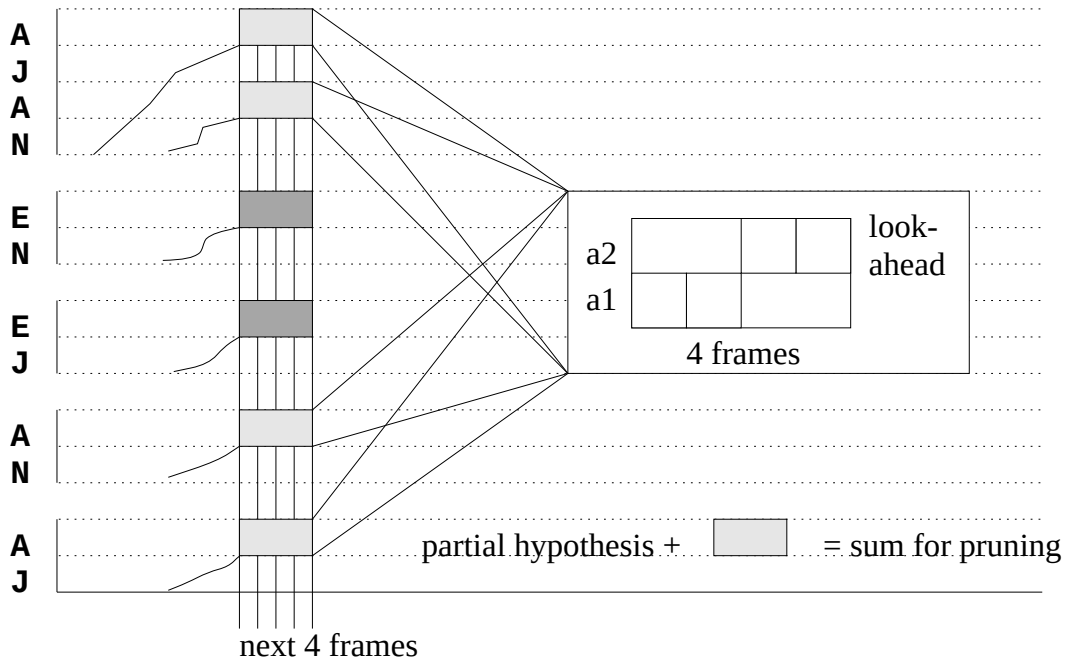


Figure 5.14: Phoneme lookaheads. In this example 6 partial hypotheses are candidates for a phoneme transition at time t_1. The sum of the cumulative score of each partial hypothesis and the lookahead score of the next monophone (grey rectangles) is used to decide which of these hypotheses should actually be considered for a transition.

For each phoneme transition, the sum of the score accumulated so far in the partial hypotheses plus the estimated score for the next phoneme in the word is compared

to a threshold. The transition is computed only if the sum is below the threshold.

A small context independent HMM with three states and a small number of Gaussians is used to compute phoneme lookaheads. To quickly estimate the score over the next couple of frames, a linear alignment is used rather than a *Viterbi*-alignment: A typical lookahead for the next four frames (x_1, x_2, x_3, x_4) for the monophone A (states A_1, A_2, A_3) would be computed to:

$$\begin{aligned} lookahead(\mathtt{A}, 4) = \quad & score(A_1, x_1) + score(A_1, x_2) + \\ & score(A_2, x_3) + score(A_2, x_4) \end{aligned} \tag{5.1}$$

To make up for the difference in size and accuracy of the lookahead HMMs and the full acoustic modeling, a multiplicative parameter γ is used.

During the search, transitions from a last state of an allophone are only considered if the cumulative score *score*(before transition) for the partial hypothesis in this state is small enough:

$$\begin{aligned} score(\text{before transition}) \quad & < \quad score(\text{best partial hypothesis}) \\ +\gamma\ lookahead(\mathtt{A}) \quad & \quad +\text{pruning threshold} \end{aligned} \tag{5.2}$$

Experiments

The *phoneme-lookaheads* used for these experiments are small, context independent *hidden Markov models*: using the labels of the best system at the time, a little HMM with 16 Gaussians per codebook and one codebook for each of the 3 states of the context independent monophones is trained. To avoid duplicate effort, exactly the same preprocessing is used for the main and for the the lookahead acoustic. However, the lookahead system only uses the first 16 coefficients of each feature vector.

The phoneme lookaheads reduce the number of active roots and nodes in the search tree. If fewer roots and nodes are active, the number of computed observation probabilities drops too, resulting in a considerable speed-up of the total system.

For these experiments with a four frame lookahead, a linear alignment between the first two HMM states and the next four input vectors was chosen: the lookahead score is the sum of the scores for the first state over the next two frames and the scores for the second state over the two frames after that. Using all three states of the lookahead HMM or using more complex alignment strategies did not yield any improvements in preliminary experiments.

Figure 5.15 shows how the number of active nodes decreases for a given pruning threshold. Even for very modest pruning thresholds that do not affect the recognition accuracy, the number of active normal nodes is reduced significantly by using phoneme lookaheads. The number of active root nodes is reduced by a factor of two.

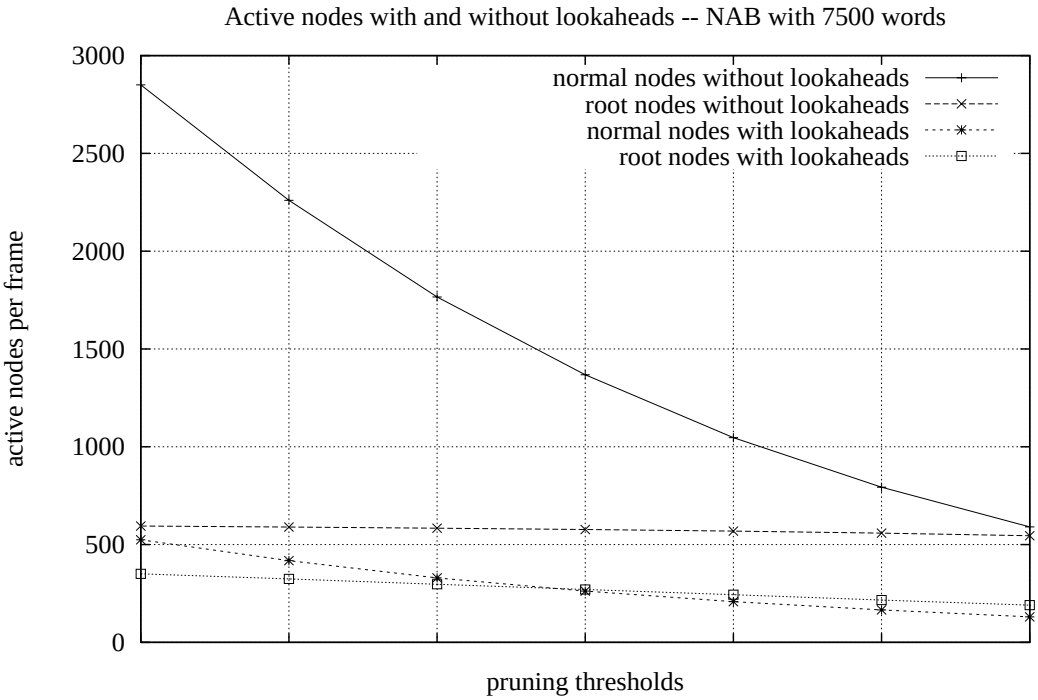


Figure 5.15: Number of active nodes with and without *phoneme lookaheads*. The pruning thresholds get tighter from left to right (on NAB using the reduced 7,500 word vocabulary).

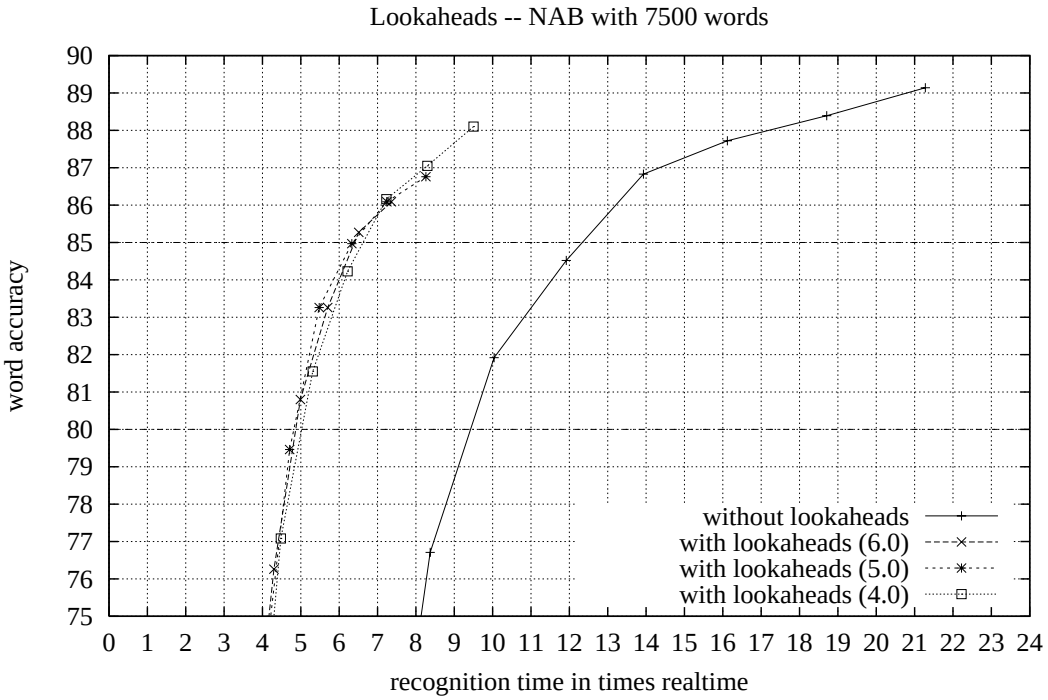


Figure 5.16: *Phoneme-Lookaheads* with different lookahead weights.

Figure 5.16 shows how the recognition time for the same word accuracy can be reduced with phoneme lookaheads for three different lookahead weights. The exact setting of the lookahead weight does not have a large impact on the performance of the algorithm. The initial lookahead-factor of 5.0 was estimated by comparing the average score per phoneme for the main acoustic model to the lookahead model. Lookahead-factors larger than 7.0 lead to a significant loss in word accuracy, while for lookahead-factors smaller than 3.0 reduce the possible speed-up. Using phoneme lookaheads, the recognition time for a recognizer using a 7,500 word vocabulary at a given word accuracy is cut in half.

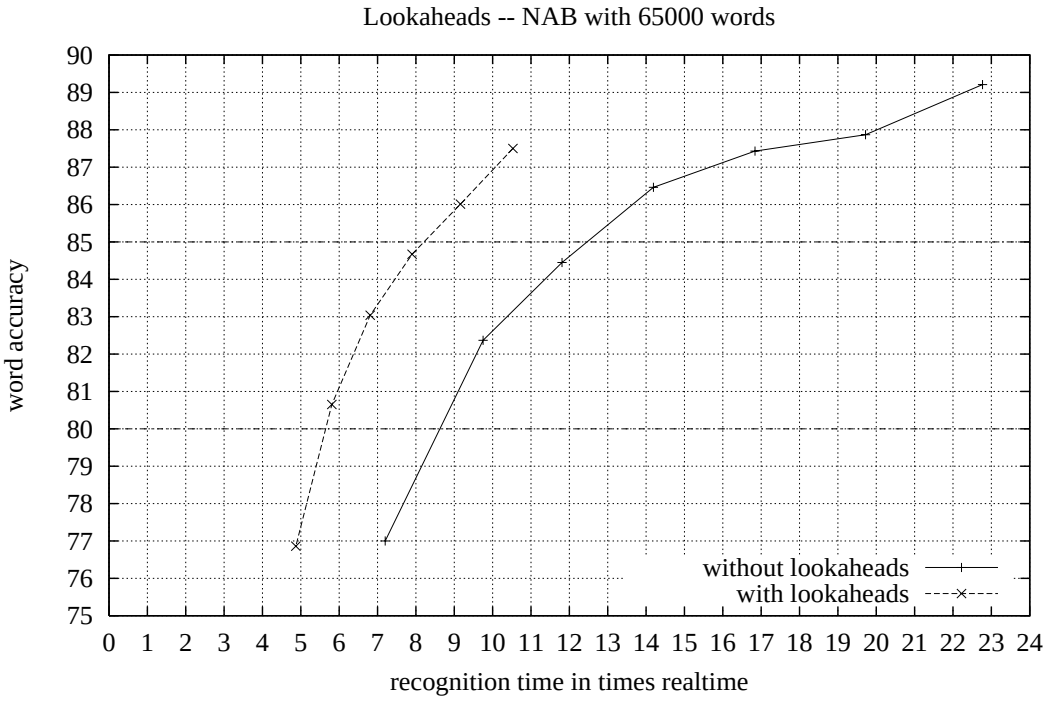


Figure 5.17: NAB *phoneme lookaheads* using the full 65,000 word vocabulary.

Figure 5.17 shows the improvement for a single lookahead weight using the full 65,000 word vocabulary. Since the larger vocabulary adds more leafs than root nodes to the search tree, and because leafs can be efficiently pruned without lookaheads, the relative speedup due to phoneme lookaheads is smaller for larger vocabularies.

5.2 Observation Probabilities

The runtime profile in figure 5.18 shows that the recognition time for the *North-American-Business-News* recognizer is dominated by the computational effort for evaluating the observation probabilities. Though the profile has been created using a simplified version of the score computation (nearest neighbor approximation), it still takes 70% to 90% percent of the total recognition time depending on the vocabulary size. This is why faster score computation algorithms play an important role in this thesis.

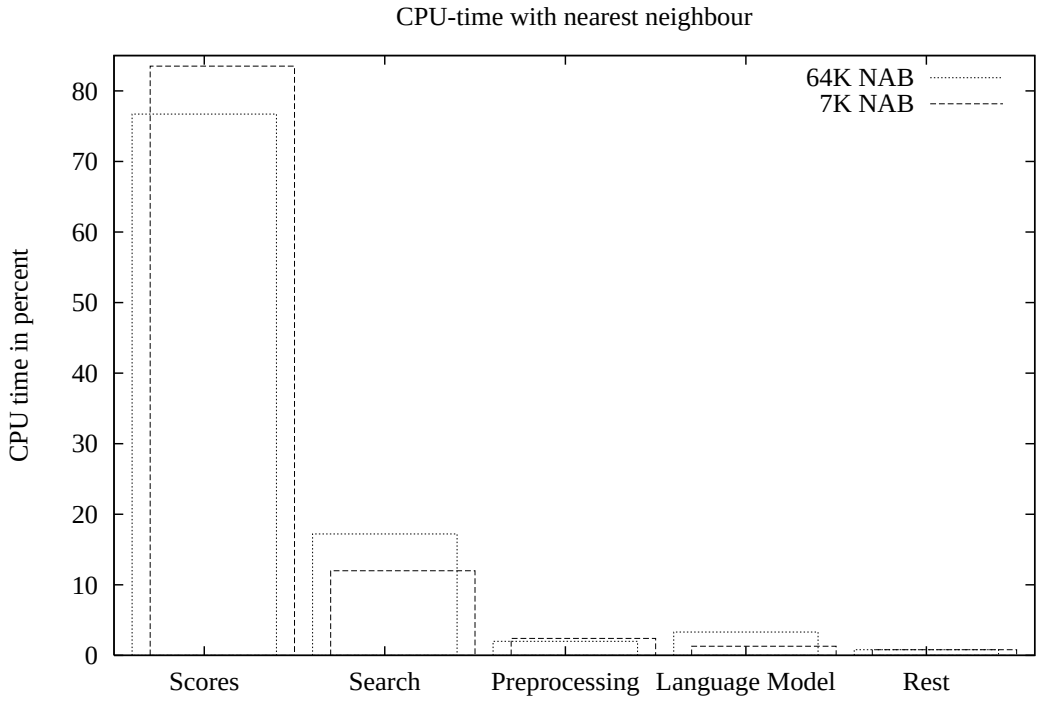


Figure 5.18: Runtime profile using the nearest neighbor approximation. Using the same acoustic modeling, the relative effort for the search grows with the vocabulary size.

Most experiments in this chapter have been performed using the *nearest neighbor approximation* and *phoneme lookaheads*. Each section in this chapter builds on earlier sections, comparing the runtime behavior of new algorithms to the behavior of the best approach so far. The following approaches will be presented in this section:

→ 77 **Order of the Gaussians in the codebook:** When using the nearest neighbor approximation, the computation of the *Mahalanobis* distance can often be abandoned after considering only a few coefficients. If the Gaussian with the smallest *Mahalanobis* distance to the input vector is used first, the computation for the subsequent Gaussians can be interrupted earlier. It is therefore beneficial to first try the Gaussians that are more likely to be the nearest neighbor.

→ 78 **Removing unused Gaussians:** If during the recognition phase a particular Gaus-

sian is hardly ever used as the nearest neighbor, it can be removed from the codebook without influencing the recognition result.

Reduced Dimensionality: The baseline system for the NAB experiments uses 48 coefficients per Gaussian. One approach to reducing the effort for finding the nearest neighbor is to only use the most important of these coefficients. → 79

Radial Covariances: Using radial rather than diagonal covariances saves one multiplication per coefficient in the computation of each *Mahalanobis* distance. → 81

Bucket Box Intersection: These experiments show that the new implantation of the *bucket-box-intersection* algorithm outperforms all other methods tried in this section, except for the Generalized-BBI. → 84

Generalized-BBI: The Generalized-BBI is an extension of the BBI-algorithm that has been developed by the author to compensate for problems that arise when applying the BBI to systems with fully continuous HMMs. The Generalized-BBI uses larger BBI-trees which can cover any number of codebooks [Woszczyna and Fritsch, 1997]. → 87

5.2.1 Order and Importance of Gaussians

Order of the Gaussians

If the scoring algorithm only uses the distance to the nearest neighbor, the computation for one Gaussian can be aborted if the distance over the first coefficients exceeds the distance to the nearest Gaussian so far. If the nearest Gaussian happens to be the first in the codebook, subsequent computations will be aborted earlier.

The order of the Gaussians within the codebook is therefore important. There is a long term and a short term optimum for this order:

1. The more frequently a Gaussian is the nearest Gaussian during the recognition phase, the earlier it should be tried.
2. The nearest Gaussian in the previous frame is most likely to be the nearest Gaussian in this frame, since the speech signal usually changes slowly.

To accommodate both requirements, the following algorithm was used to update the order of the Gaussians in the codebook: for every score computation, the nearest Gaussian is moved from whatever position it had in the codebook to the first position, satisfying the short term consideration. Moving the best vector to the first position also performs a kind of speaker adaptation, moving the vectors required for the current speaker to the front. The remaining Gaussians move down to fill the resulting gap. Frequently used Gaussians will thus remain among the first Gaussians to be tried in the score computation. The overhead of this method is almost zero, since the implementation of the score computation in the JRTk already uses a list

containing the indices of the Gaussians to use in a score computation. The effort of rearranging the indices in this list is very small compared to the actual score computation.

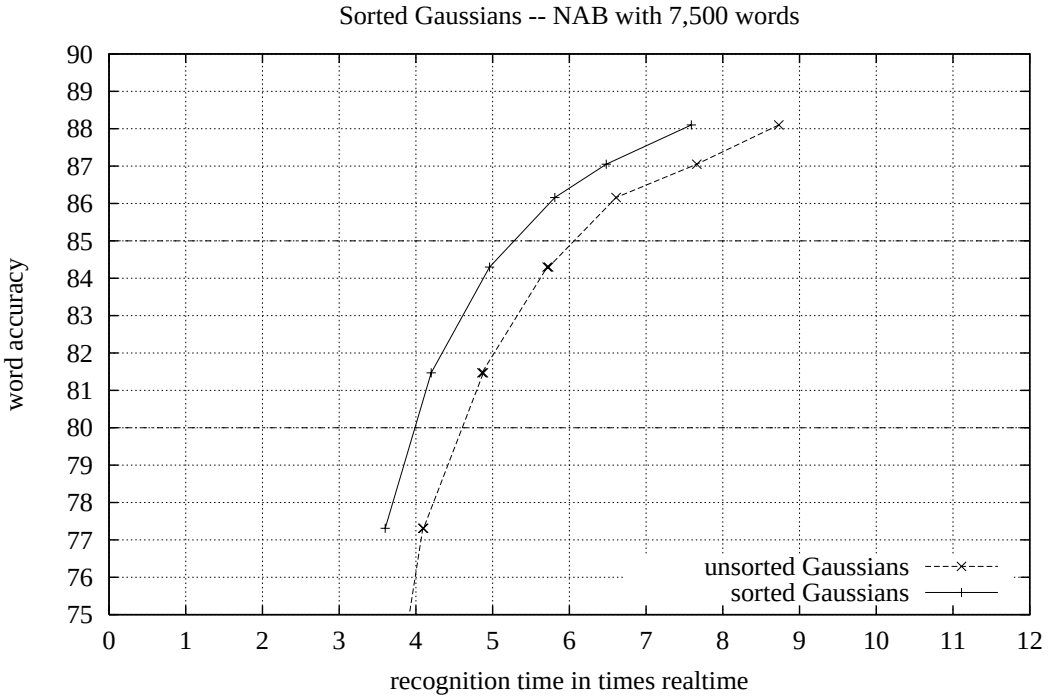


Figure 5.19: Reordered vectors

Figure 5.19 shows the speed-up due to using the resorted Gaussian list. Note that keeping a counter for each Gaussian to order them by their actual frequency is not only much more expensive but also less effective since the short term importance of the Gaussians is neglected.

Removing Unused Gaussians

When using the nearest neighbor approximation, some of the Gaussians contribute only rarely to the score computation. While completely removing those Gaussians reduces the word accuracy, there is also a considerable speed-up. In this experiment a Gaussian is removed from the codebook if it is used 100 times less frequently than the most frequently used Gaussian of the same codebook.

Figure 5.20 shows the decrease in word accuracy and the increase in recognition speed due to removing rare Gaussians.

5.2.2 Reduced Dimensionality

For some systems a *linear discriminant analysis* (LDA) has become an integral part of the preprocessing. The goal of the LDA is to provide feature vectors with

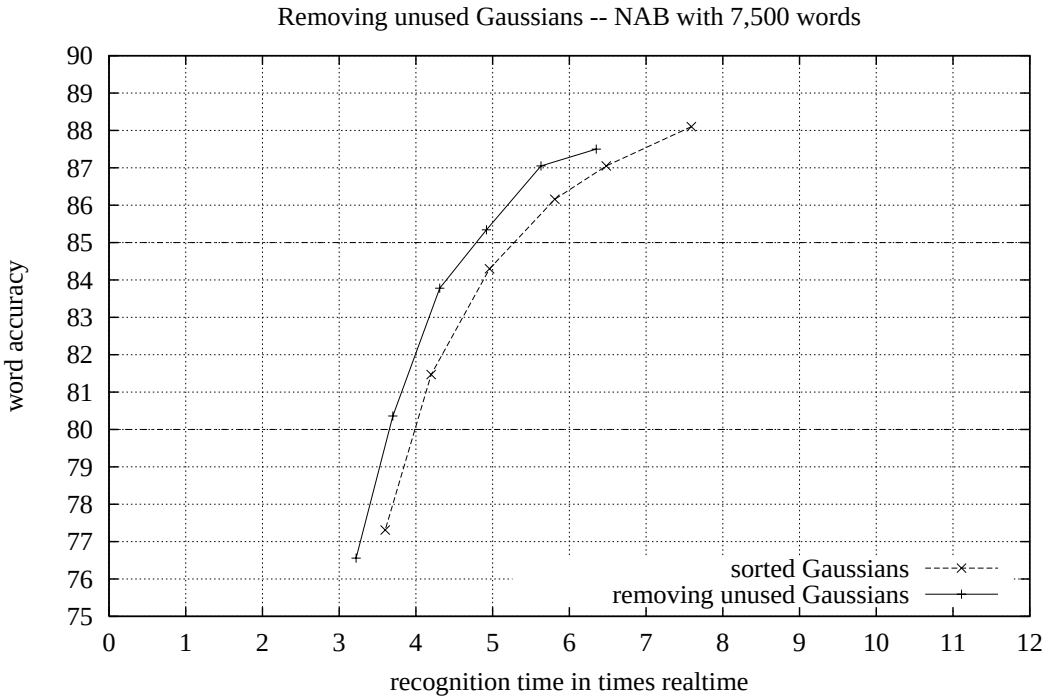


Figure 5.20: Removing rare Gaussians

unrelated coefficients, which are sorted by variance. Coefficients with a higher index can be assumed to be less important for the recognition compared to coefficients with smaller indices.

It is therefore possible to use only a subset of the coefficients to determine the nearest neighbor, and then calculate the exact distance to this Gaussian over all coefficients.

Experiments

The baseline system for the NAB experiments uses 48 coefficients per Gaussian. Due to the linear discriminant analysis in the preprocessing, the variance increases with the order of the coefficient. It can therefore be assumed that some of the higher coefficients contain mostly noise and contribute little to the choice of the nearest vector.

The effort for retraining a new system with fewer coefficients is usually too large. Also, the absolute size of the scores depend on the number of coefficients used. Therefore, a reduced number of coefficients is used only for the most costly step in the score computation, namely the task of identifying the nearest neighbor. The actual score is then computed using all 48 coefficients, keeping the size and accuracy of the scores comparable to the original system.

Figures 5.21 and 5.22 show the influence of using 36 instead of 48 coefficients on a system with 7,500 and 65,000 words respectively.

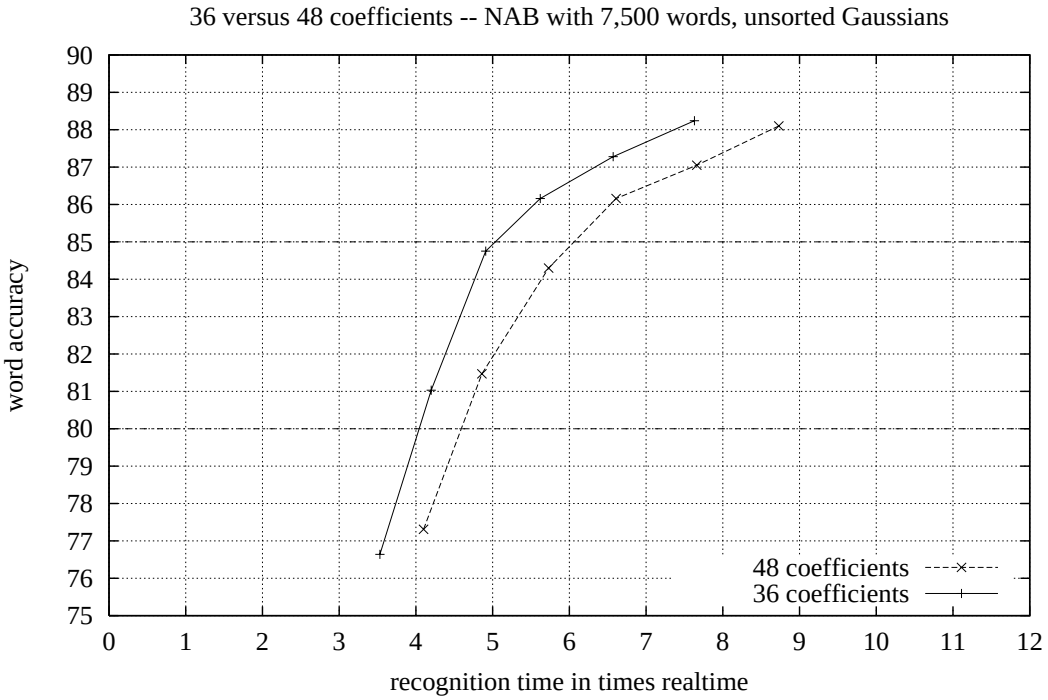


Figure 5.21: Using 36 instead of 48 Coefficients to determine the nearest neighbor on NAB with 7,500 words.

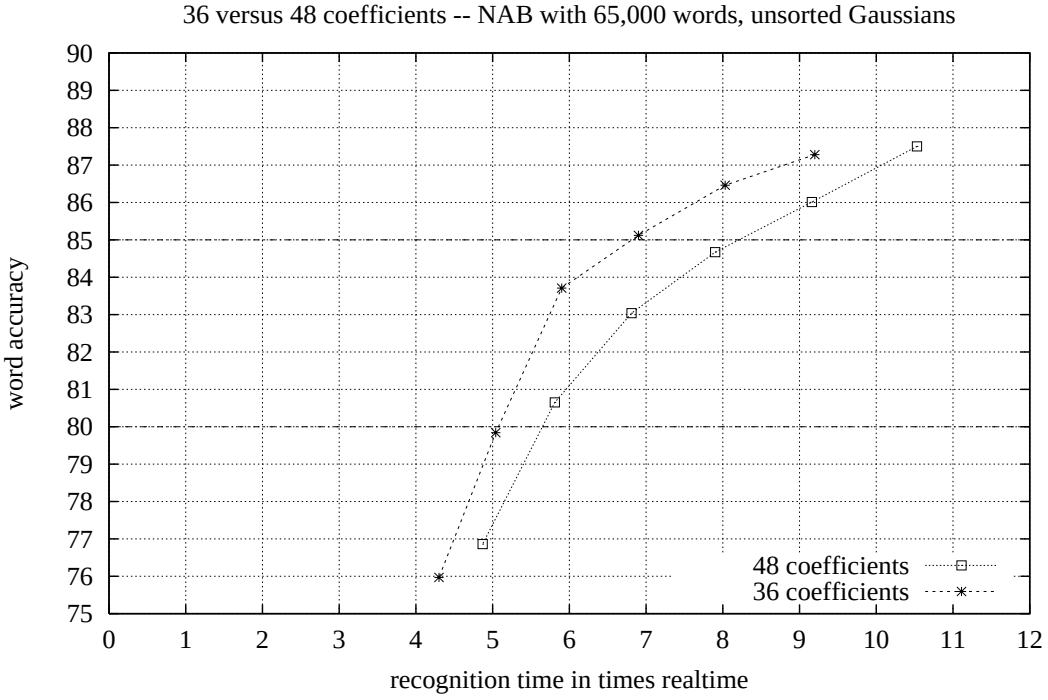


Figure 5.22: Using 36 instead of 48 Coefficients to determine the nearest neighbor on NAB with 65,000 words.

These experiments were performed before the introduction of the Generalized-BBI algorithm. Using the Generalized-BBI algorithm, the number of Gaussians from which the nearest neighbor has to be chosen is significantly smaller. Therefore, the speed-up when using a smaller number of coefficients to determine this Gaussian is insignificant when combined with the Generalized-BBI.

5.2.3 Radial Covariances

Using radial rather than diagonal covariances saves one multiplication per coefficient in the computation of each *Mahalanobis* distance. However, the resulting speed-up on modern processors with several FPUs and long pipelines is much smaller than expected. Also, the accuracy of the resulting scores is smaller, shifting the load from the score computation to the search process. The remaining overall speed-up is not significant.

5.2.4 Low Level Optimizations

If an inefficient implementation of an otherwise correct algorithm is considered a bug, this paragraph is about workarounds for bugs. These bugs can often be considered to be bugs in the c optimizer rather than in the implementation of the score computation: depending on the compiler, the optimizer and the processor architecture, measures like unrolling (SUN) or not unrolling (ALPHA) loops, using floats instead of doubles (SUN) or doubles instead of floats (ALPHA) and slightly rearranging the data structures can reduce the total runtime by 20% to 30%. Since attempts to work around these optimizer problems seriously affect the readability and portability of the resulting code they have been strictly limited to the innermost loops of the most expensive routines. All experiments up to the section on BBI have been performed without low level optimizations. Starting with the BBI section, these improvements are always included.

5.2.5 Bucket Box Intersection

The *bucket-box-intersection* (BBI) algorithm [Fritsch and Rogina, 1996] is a generalized version of the *bucket-Voronoi-intersection* algorithm [Fritsch et al., 1995] [Ramasubramanian and Paliwal, 1992a].

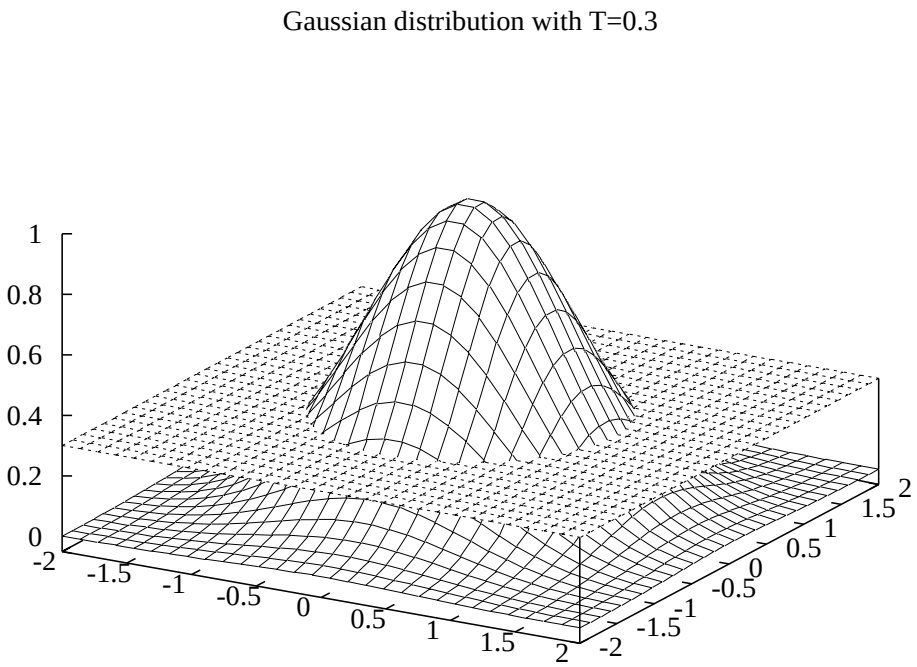


Figure 5.23: *Bucket Box Intersection*: the Gaussian is cut off at T=0.3; the rectangle around the intersection of the Gaussian with the plane defines the border of the Gaussian box.

For each codebook vector, a rectangular box is defined around the ellipsoid where the value of its Gaussian distribution falls below a threshold T (figure 5.23). If the input vector $\mathbf{x}$ does not fall into the Gaussian box of a codebook vector, this codebook vector can be ignored when computing the observation probability for $\mathbf{x}$. The maximum error of this approximation is given by the value of T.

To quickly eliminate Gaussian boxes that do not contain an input vector, the BBI uses a pre-computed binary decision tree over each codebook to dynamically determine the subset of Gaussians to be used in the score computation. This tree is a K-dimensional space partitioning tree (K-d tree) developed by Bentley [Bentley, 1975] is used. Each question of the decision represents a hyper-plane in K-dimensional space perpendicular to one axis. Such a hyper-plane can be described by the intercept of the hyper-plane with this axis. If the input vector lies to one side of the hyper-plane, then it cannot fall into any Gaussian boxes that are completely on the other side of this hyper-plane.

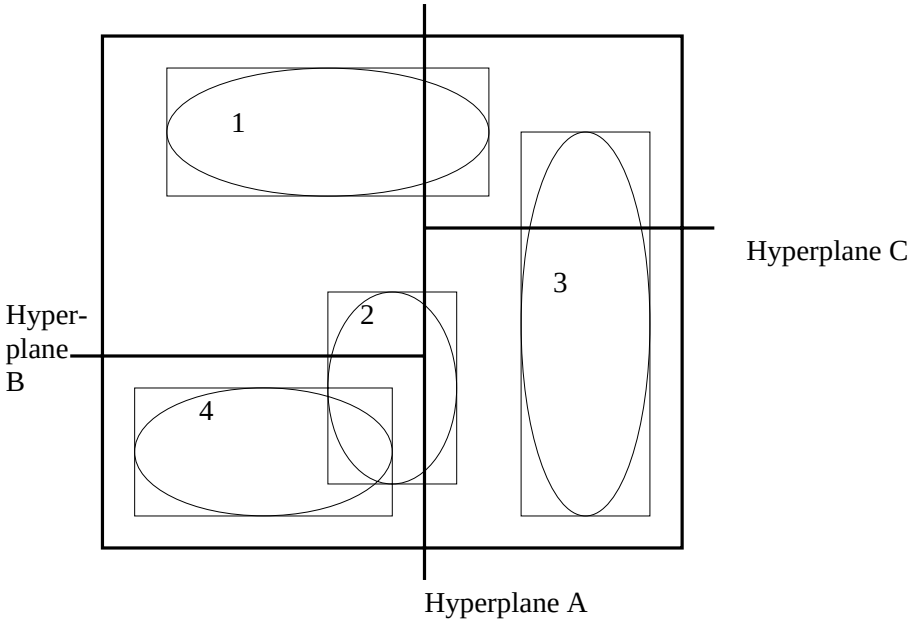


Figure 5.24: Bucket Box Intersection example in two-dimensional space.

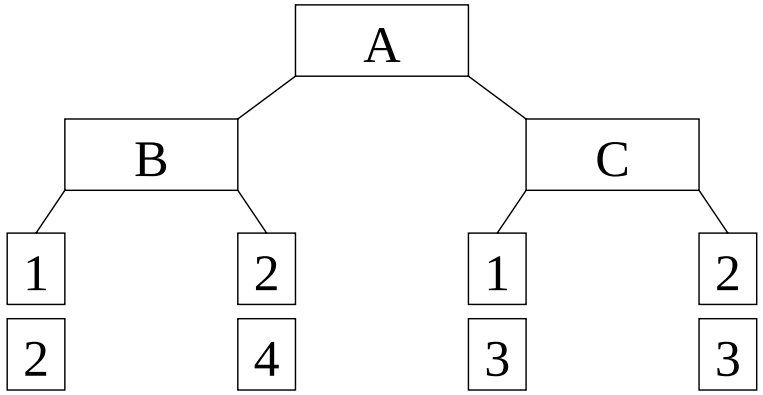


Figure 5.25: *Bucket-Box-Intersection* tree.

The BBI-trees for a speech recognition system (one per codebook) are pre-computed after the training of the recognizer. At the beginning of the tree building algorithm there are no deviding hyper-planes and the Gaussians of all vectors share one node. The next step is to find the optimum hyper-plane to split this node. For each axis an experimental hyper-plane is determined perpendicular to the axis such that the same number of Gaussians in the node are on either side of the hyper-plane. Of those hyper-planes, the one that intersects with the fewest Gaussian boxes is then chosen to divide this node. Gaussian boxes to one side of the hyper-plane go into one node, Gaussian boxes on the other side of the hyper-plane into the other node, and Gaussian boxes that intersect the hyper-plane go into both nodes. This procedure is repeated until a predefined tree depth is reached. The final nodes that contain one or several Gaussians each are called *buckets*.

While this algorithm does not produce an optimal tree with respect to the test

situation,[2] the resulting tree is well balanced and can be built within less than one hour based on the systems codebooks only.

The BBI algorithm yields the best improvement for systems that have few but large
→ 23 codebooks and is therefore especially well suited for semi-continuous HMMs.

Experiments

The experiments in this section have been performed after the introduction and
→ 87 preliminary experiments for the Generalized-BBI. The original intent of the experiments was to show the benefits of the Generalized-BBI over the normal BBI presented here. Therefore, the *phoneme-lookaheads* used in this section are already computed using a single Generalized-BBI tree. Since the overall contribution of the scores for the lookaheads to the computational load is low, this has little impact on the total computation time.

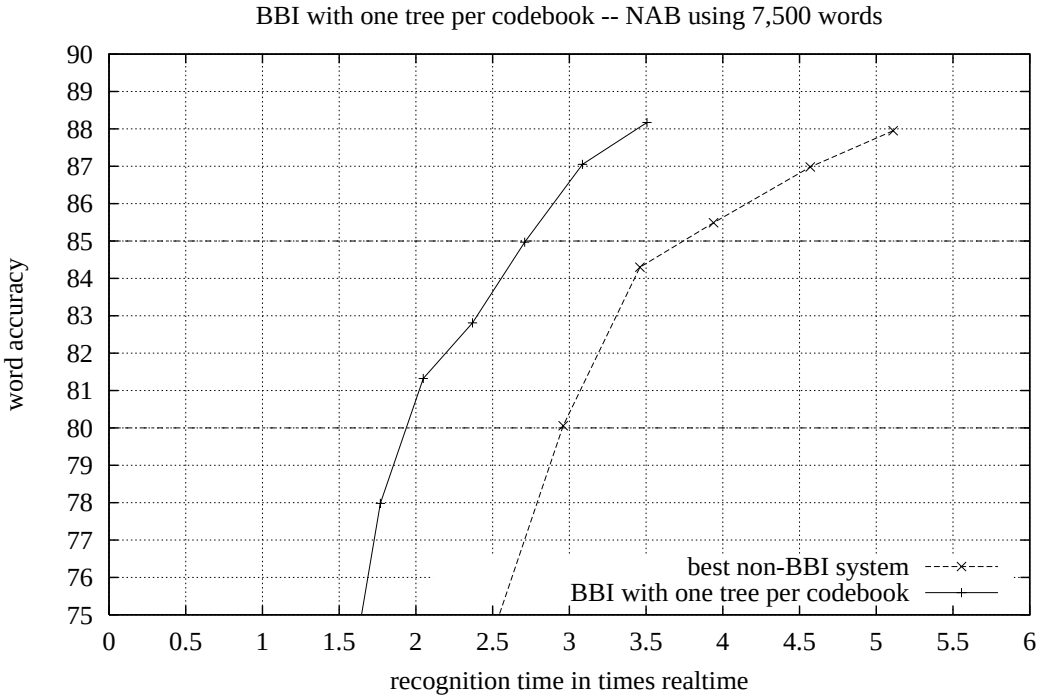


Figure 5.26: Normal BBI

Figure 5.26 shows that using normal BBI with one small BBI tree for each of the 3,000 codebooks of the system results in a 20% speed-up over a system using the fastest score computation method so far. The experiment was performed using a tree depth of 6, which was found to be optimal for a cross-validation set. The Gaussian boxes were chosen to intersect the Gaussians at 50% of their maximum value. The
→ 45 BBI was used in conjunction with the *Nearest Neighbor Approximation*, using only

[2] An optimal BBI-tree would be tuned to on average return buckets with as few Gaussians as possible during a normal recognition run.

the Gaussian in the selected BBI bucket that has the smallest *Mahalanobis* distance to the input vector to estimate the observation probabilities.

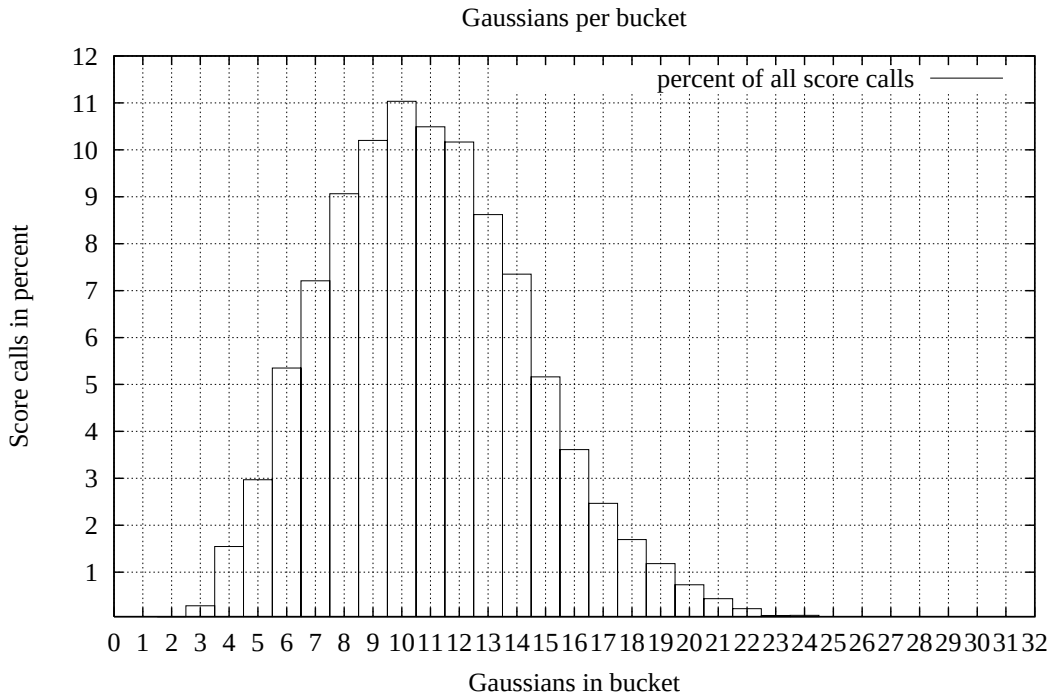


Figure 5.27: Different *buckets* of the BBI tree contain different numbers of Gaussians. This plot shows the percentage of observation probabilities that are estimated using a bucket that contains a given number of Gaussians.

How the BBI-speed-up is achieved is illustrated in figure 5.27: since the average number of Gaussians in the selected BBI-bucket for a corresponding codebook is about ten, the average number of *Mahalanobis distances* to be computed drops from 32 to 10.

BBI-Summary

By combining the different techniques to speed up the score computation within each codebook (nearest neighbor approximation, code optimizing, BBI), the total recognition time can be reduced by a factor of four.

Figures 5.28 and 5.29 show the respective improvements for a 7,500 word vocabulary and a 65,000 word vocabulary.

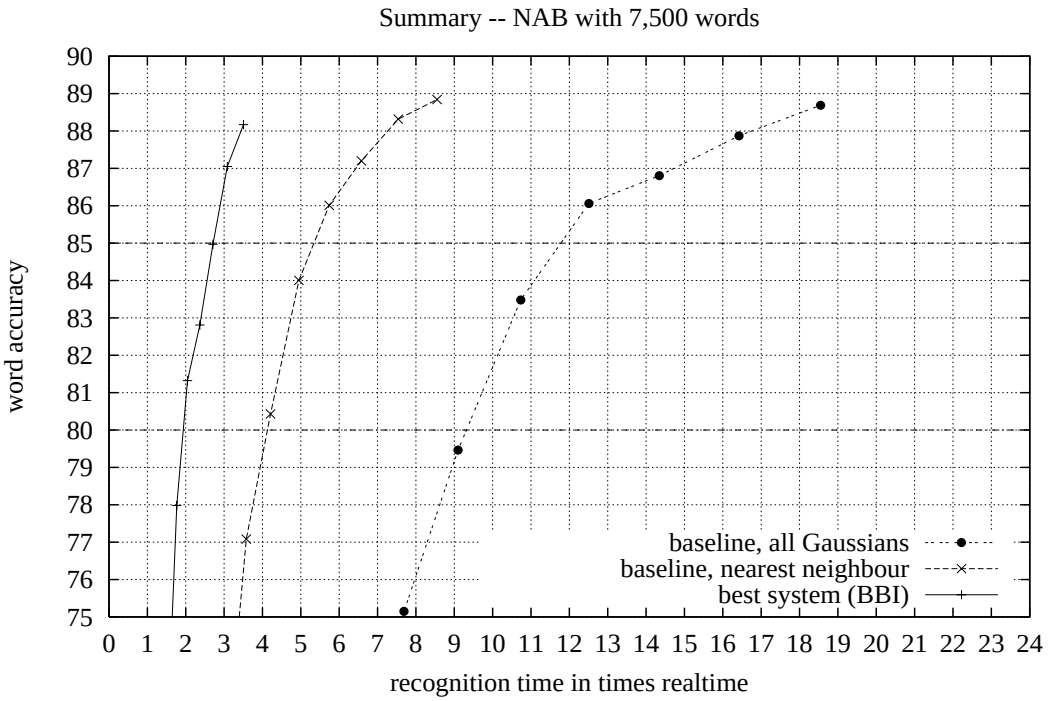


Figure 5.28: Recognition time with/without combined improvements on NAB with 7,500 words in the vocabulary.

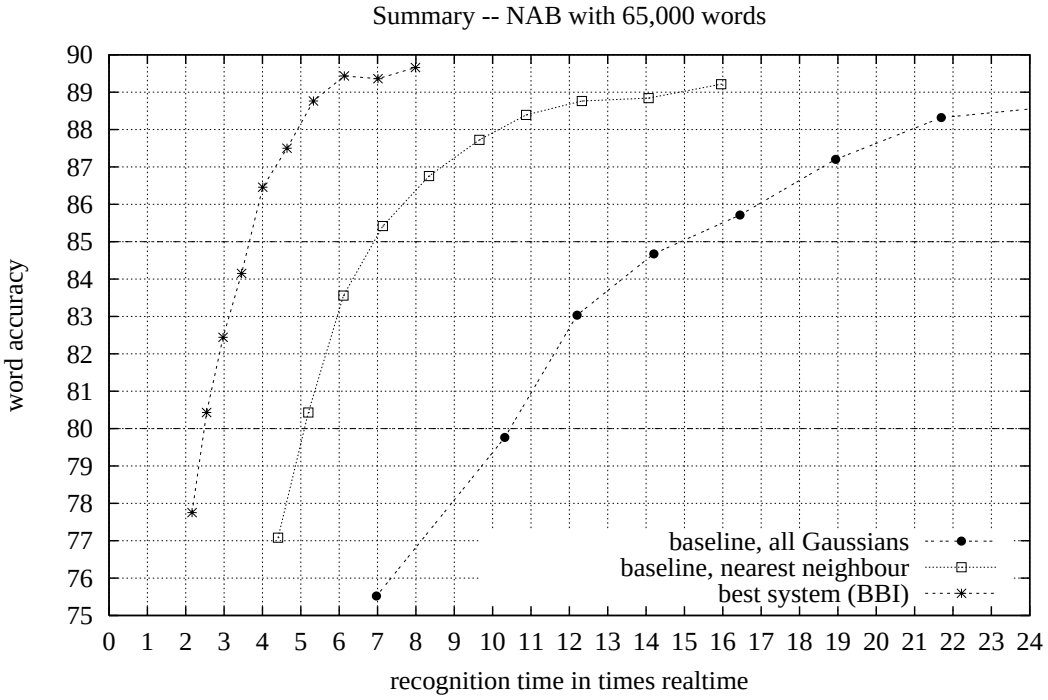


Figure 5.29: Recognition time with/without combined improvements on NAB 65,000 words

5.2.6 Generalized-BBI

Most of the large vocabulary continuous speech recognition systems built with the JRTk use thousands of small codebooks to model the sub-allophones. For this configuration, the *Bucket-Box-Intersection* algorithm is not well suited because the number of Gaussians over which each binary tree can be built is very small. → 82

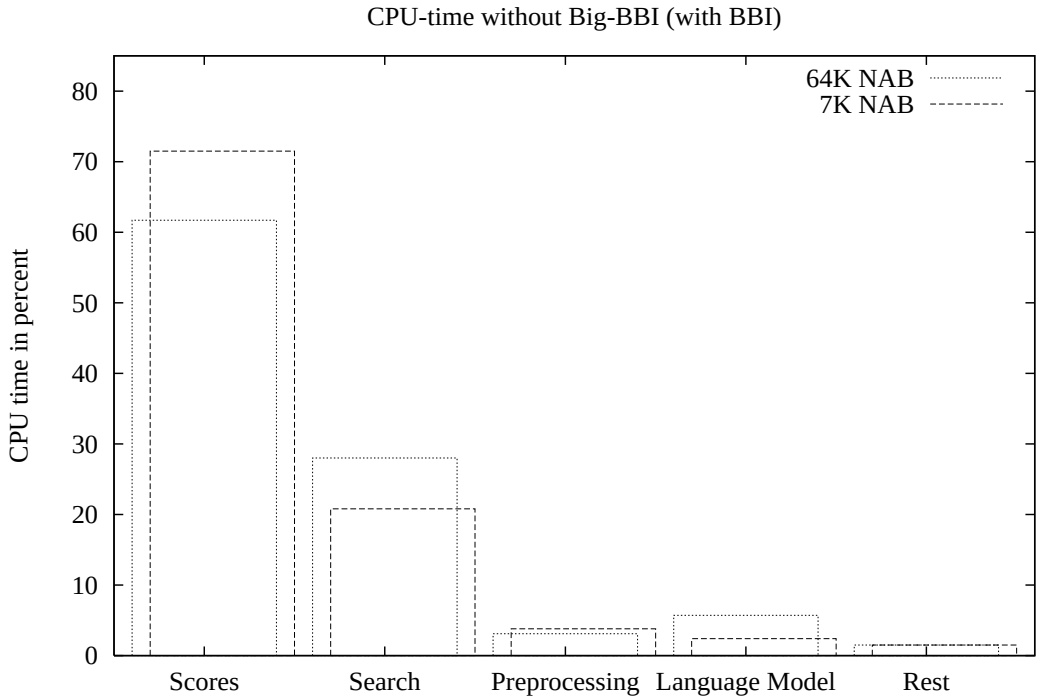


Figure 5.30: Runtime profile with normal BBI

As can be seen in figure 5.30, even with a combination of the BBI algorithm and the *Nearest Neighbor Approximation*, the score computation still requires a significant portion of the total CPU-time for the NAB system used.

The average number of scores computed per frame is about 200 to 400, but only one of these scores contributes to the globally best hypothesis. Most of the score computations are for allophones that are not closely related to the signal observed at the moment. The distance from the input vector to all Gaussians in the codebooks associated with these allophones is usually large. Therefore, the score for these allophones is not likely to contribute to the best hypotheses and the influence of the correct choice of the codebook vectors used for these score computations is small.

To take advantage of this, for the Generalized-BBI algorithm [Woszczyna and Fritsch, 1997] larger BBI trees are used, which can cover any number of codebooks. In the most common configuration, a single Generalized-BBI tree is built over the Gaussians of all codebooks (Big-BBI). To estimate the observation probability for an allophone that is modeled by a codebook which is not represented by a Gaussian in the bucket associated with the input vector, a

back-off strategy has to be used. In this case all Gaussians of this codebook have a large distance to the current input vector.

For the initial experiments, the Gaussian in the codebook that got the most training was chosen as a back-off. Also, for non-back-off score computations, all Gaussians in the bucket were initially used to estimate the observation probabilities. Later in this chapter it will be shown how the performance of the Generalized-BBI can be improved by combining the Generalized-BBI with the *Nearest Neighbor Approximation* and by using more sophisticated back-off strategies.

Generalized-BBI with all Gaussians

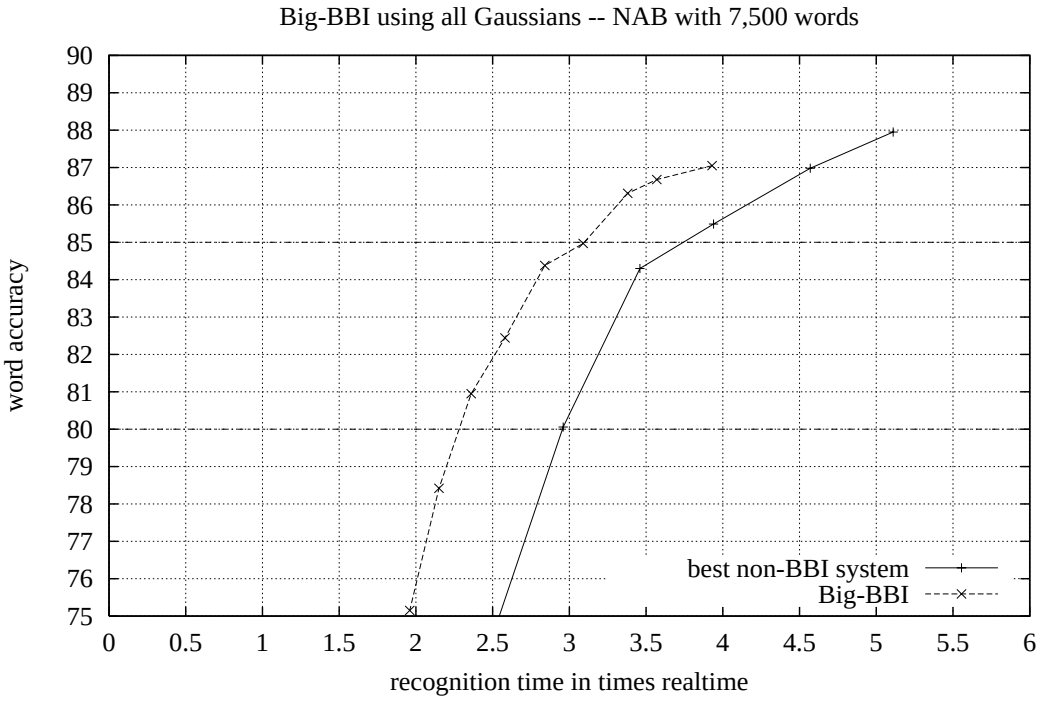


Figure 5.31: Big-BBI using all Gaussians on NAB with 7500 words

Figure 5.31 shows that though there is a pronounced loss in word accuracy, the Generalized-BBI also allows a higher recognition speed compared to the best system so far.[3]

Figure 5.32 illustrates the difference in the speed-up mechanism between the BBI and the Generalized-BBI algorithm: Using the BBI algorithm the speed-up is achieved by a uniform reduction of the number of Gaussians per score call over all codebooks
→ 85 (figure 5.27) regardless of the distance of the codebook to the input vector. For

[3] These results should not be compared to the BBI-system presented in the previous section, because they have been produced without the *Nearest Neighbor Approximation* and using an older code version. A comparison between Big-BBI and normal BBI is shown in figure 5.38.

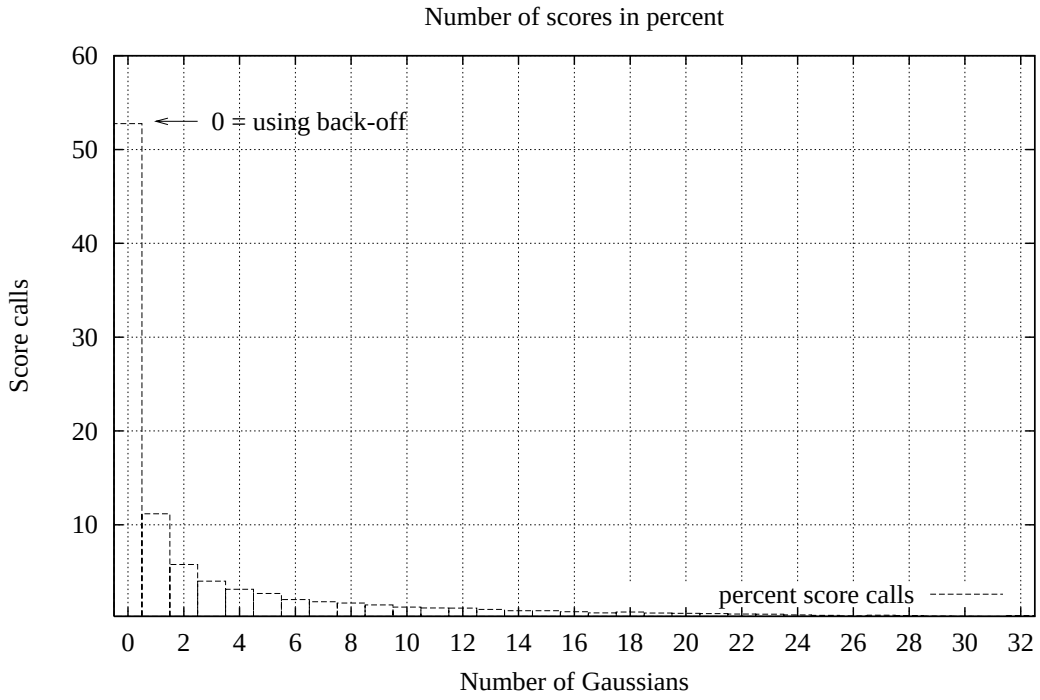


Figure 5.32: Distribution of the score calls over the number of Gaussians in the selected Generalized-BBI bucket.

the Generalized-BBI algorithm, the scores for most codebooks are estimated using a back-off value or a single Gaussian of that codebook, while the scores for the few codebooks that are close to the input vector are computed accurately with many Gaussians.

As can be seen in figure 5.32, the total number of *Mahalanobis* distances computed for the above experiment is largest for the buckets containing two or more Gaussians. This is because for a bucket with two Gaussians, two distances need to be computed, and for a bucket with 32 Gaussians, 32 distances are required. It should therefore be possible to obtain an additional speed up by combining the Generalized-BBI with the *Nearest Neighbor Approximation* to reduce the computational effort in buckets with large numbers of Gaussians.

One of the next questions that needs to be answered is whether the loss in word accuracy is due to a poor choice of the Generalized-BBI Gaussians, or due to a poor estimation of the back-off Gaussians.

For more than 50 percent of all score calls a back-off Gaussian (0 Gaussians in bucket) is used. This is not unexpected, since most of the 200 score calls for a given frame are for partial hypotheses that do not match the current acoustics, and will be pruned away later on; many of the corresponding codebooks do not overlap with the bucket box interval selected by the input vector. Since the back-off score calls are frequent, a good estimation of the back-off values is important because they contribute significantly to the structure of the search space.

Generalized-BBI with Nearest Neighbor Approximation

To combine the Generalized-BBI with the *Nearest Neighbor Approximation*, from all Gaussians in the Generalized-BBI bucket only the one Gaussian with the smallest *Mahalanobis* distance to the input vector is used for the actual score computation. As shown in figure 5.33 the *Nearest Neighbor Approximation* can be combined successfully with the Generalized-BBI.

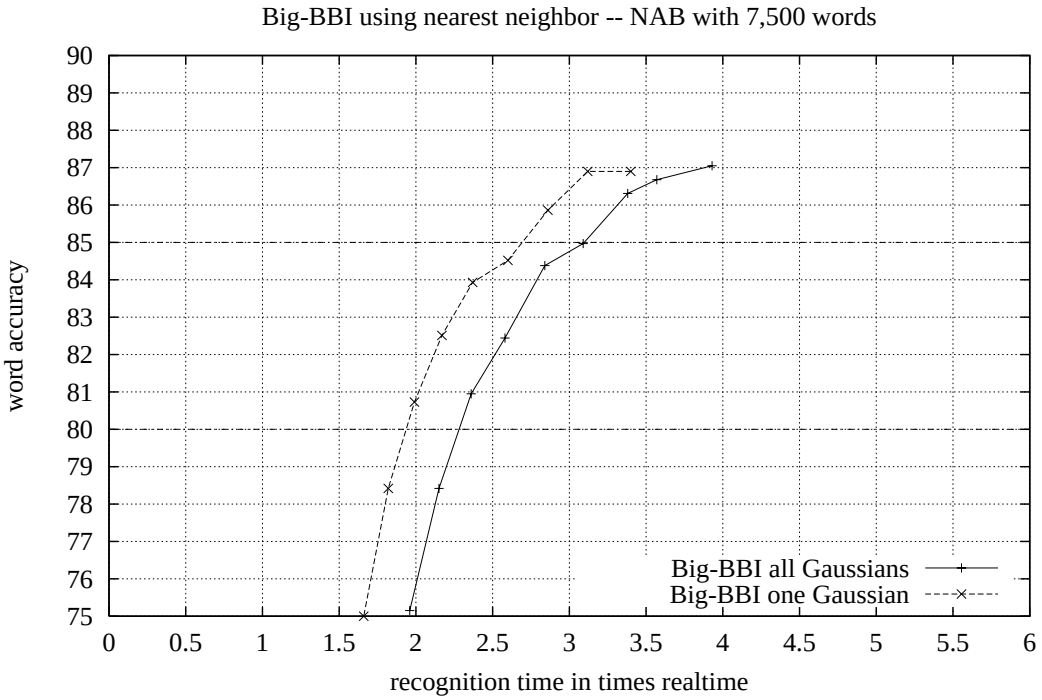


Figure 5.33: Generalized-BBI using the *Nearest Neighbor Approximation*

Generalized-BBI using Better Back-Offs

The original choice of the backoff vector for a codebook was completely independent of the current input vector. A better approximation makes the choice of the backoff vector dependent on the input vector, by having one backoff vector per codebook for each BBI bucket.

For this experiment, the Gaussian of the codebook that has the smallest distance to a bucket is included in that bucket. The metric used to measure the distance is the Euclidean distance between the Gaussian box of the Gaussian and the border of the bucket. By doing this for each codebook bucket combination, the back-off case is eliminated in the score computation.

As shown in figure 5.34, the better back-offs increase the word accuracy substantially.

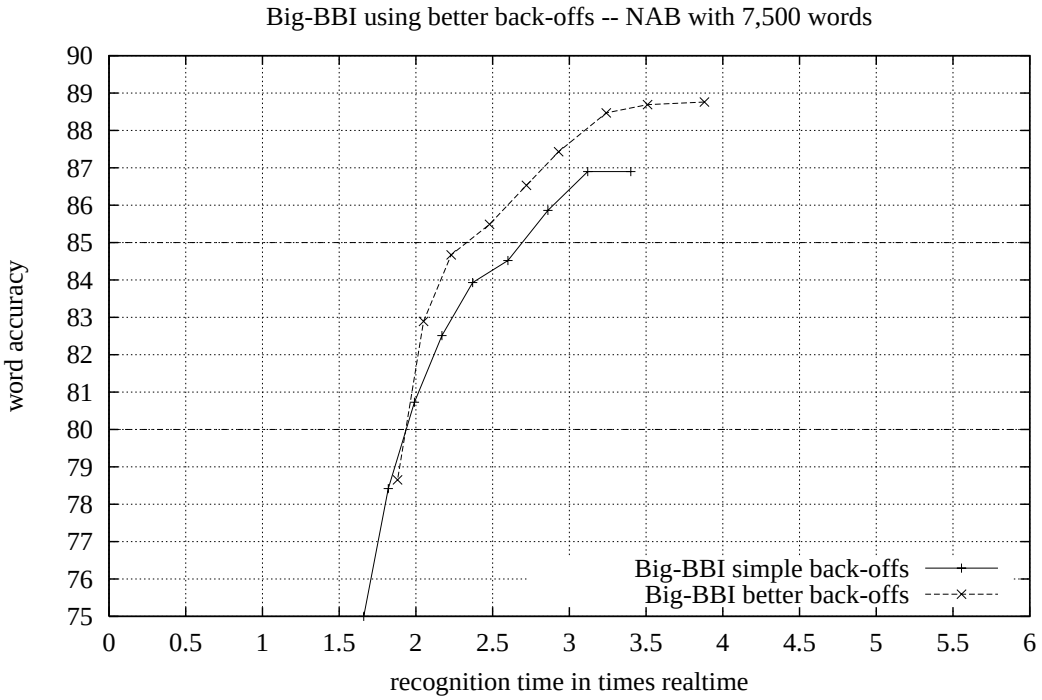


Figure 5.34: Generalized-BBI (one Big-BBI tree) with back-offs

Alternative back-off strategies

Since these initial results on improved back-off strategies for the Generalized-BBI are very encouraging, there is still ongoing research to find even better back-off strategies.

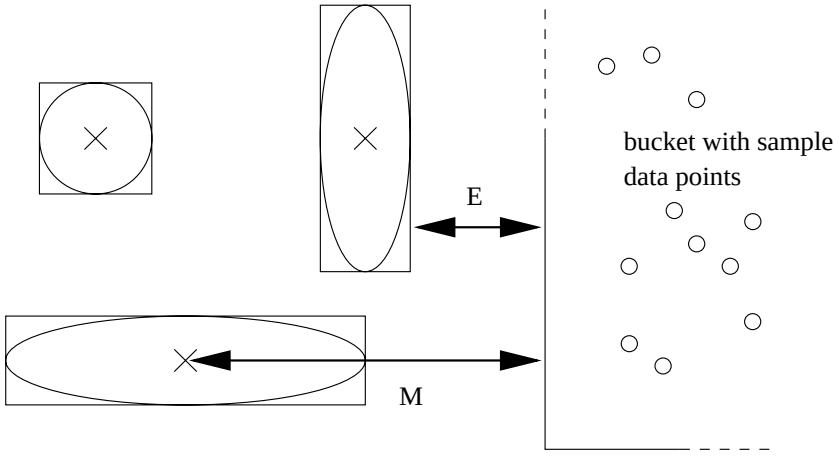


Figure 5.35: Generalized-BBI: backoff strategies

The following strategies are being explored:

1. From all Gaussians in the codebook, use the one that saw most training data.

Using this technique the resulting word accuracies are substantially smaller than the ones of the baseline system without Generalized-BBI.

2. From all Gaussians in the codebook, use the Gaussian that has the smallest Mahalanobis distance to the border of the selected bucket. In the example in figure 5.35 this algorithm would select the Gaussian marked M. The advantage over the first method is that the choice of the back-off vector depends on the selected bucket. Since the Mahalanobis distance to the border of a bucket is often only defined in one dimension, the choice of the back-off Gaussian depends mostly on the value of the covariance matrix for this direction. Experiments have shown that this method does not work significantly better than the first method.

3. From all Gaussians in the codebook, use the one that has the smallest Euclidean distance between the border of the associated Gaussian box and the border of the bucket. In the example in 5.35 this algorithm would select the Gaussian marked E. While lacking a mathematical foundation, this method was found to work substantially better than the first two approaches and is currently implemented as the standard back-off strategy.

4. Instead of the Euclidean distance to the Gaussian box, a Gaussian representing the average of all Gaussians falling into the bucket is computed. To determine the back-off Gaussian, the Mahalanobis-distance to this average Gaussian is used.

5. The back-off vector can be estimated with either of the last two methods, but if the distance of the selected Gaussian to the bucket exceeds a threshold, a constant back-off score depending only on the codebook and the selected bucket is used instead. As the distances to input vectors that fall into remote buckets tend to be error prone, the hope is to obtain a small speed-up combined with a more robust score estimation.

While some back-off strategies may seem intuitive in a two dimensional picture, the behavior in a 48-dimensional feature space partitioned by a BBI tree with a maximum depth of 10 is not easily predictable.

Generalized-BBI Parameters

For BBI and Generalized-BBI trees alike, two parameters have to be adjusted: the depth of the BBI-tree and the threshold γ for the Gaussian boxes.

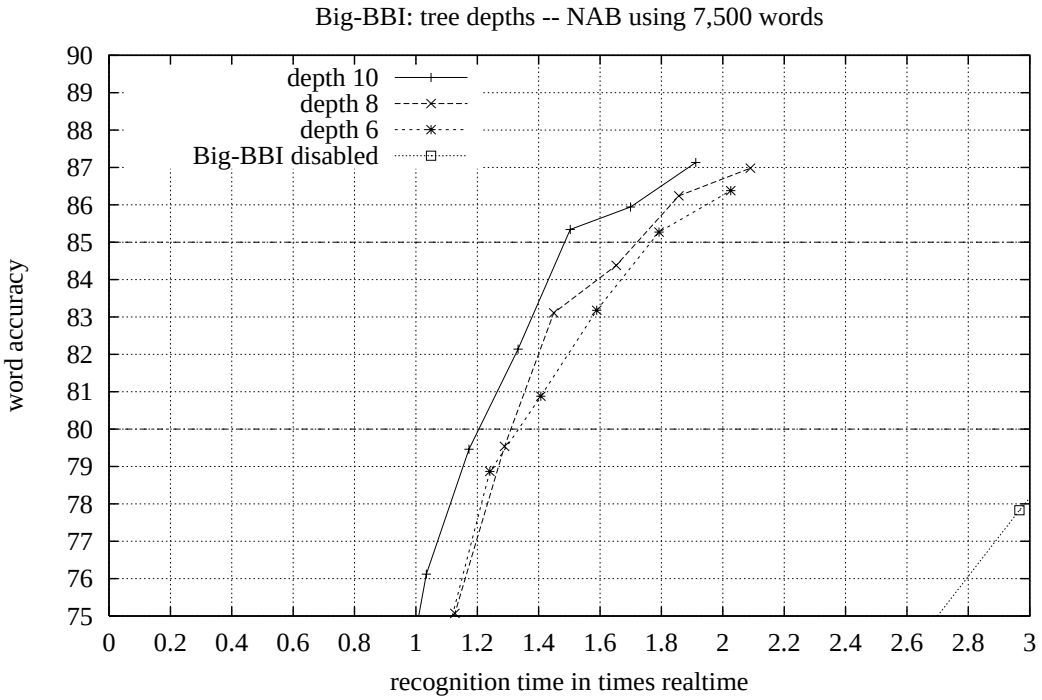


Figure 5.36: Generalized-BBI: depth of the BBI tree

Figures 5.36 and 5.37 give the word accuracy over the recognition time for a number of Generalized-BBI trees of different depths and thresholds.[4]

For smaller values of γ, the Gaussian boxes are larger. This means more boxes intersect with a bucket, and while the computational effort goes up, the word accuracy increases along with the accuracy of the probability estimation.

Deeper Generalized-BBI trees result in smaller buckets containing fewer Gaussians. While the recognition time goes down, the size of the tree grows. Since each bucket must contain at least one back-off Gaussian for each of the 3,000 codebooks, the size of the BBI-trees quickly becomes prohibitive beyond a tree depth of 10.

While it is also possible to build BBI trees over any combinations of codebooks, such as using one tree for all codebooks of one monophone, no further speed-ups or improvements could be achieved so far over just using one big BBI tree over all codebooks (Big-BBI).

[4]Note that the increase in speed over the results in figure 5.34 has two reasons: since the first results on Generalized-BBI were very encouraging, the preliminary implementation was replaced by a better and more efficient implementation. Also, the Generalized-BBI algorithm is now used for the codebooks of the phoneme lookahead models too.

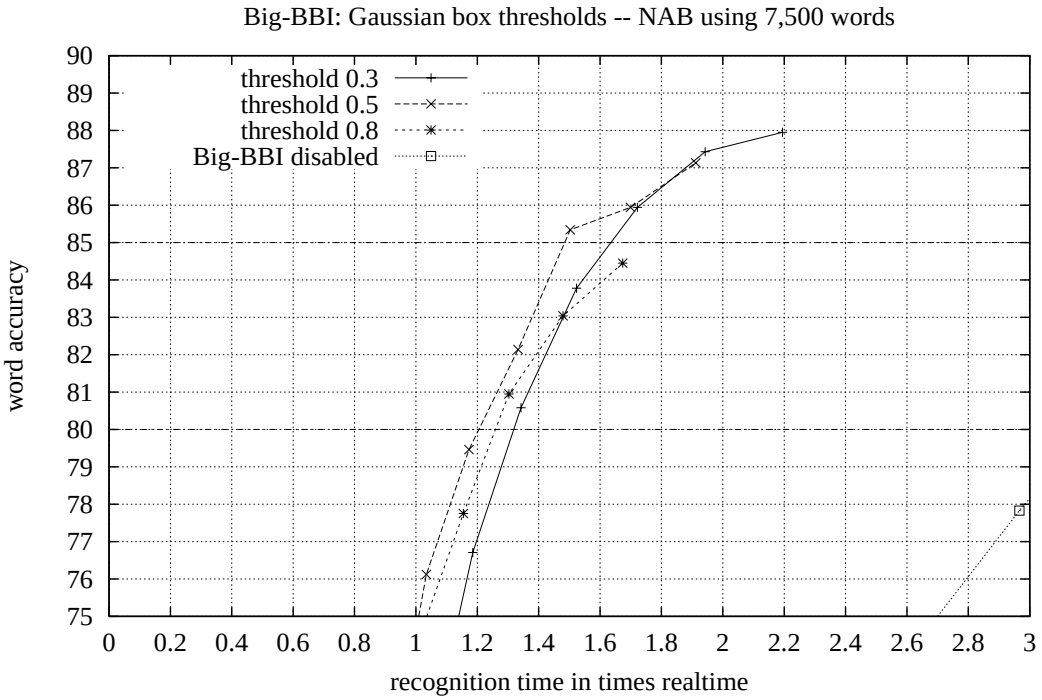


Figure 5.37: Generalized-BBI: threshold γ

depth	γ	Gaussians/bucket	Size (MBytes)	RT (WA=80)
6	0.30	13.5 %	4.3	1.65
8	0.30	8.5 %	12.6	1.45
10	0.30	5.7 %	40.8	1.30
6	0.50	9.2 %	3.3	1.50
8	0.50	5.0 %	9.7	1.30
10	0.50	2.9 %	31.7	1.20
6	0.80	4.8 %	2.4	1.30
8	0.80	2.0 %	7.3	1.30
10	0.80	0.9 %	25.7	1.25

Table 5.2: Comparison of different Generalized-BBI trees

Generalized-BBI Summary

For a vocabulary of 7,500 words, using Generalized-BBI instead of normal BBI reduced the recognition time by 35% from 1.9*RT to 1.2*RT at a word accuracy of 80%.

The effort to add the Generalized-BBI to an existing system is very small: the only thing that has to be done is the computation of the Generalized-BBI tree, which takes about one hour on the SUN-Ultra used for these experiments.

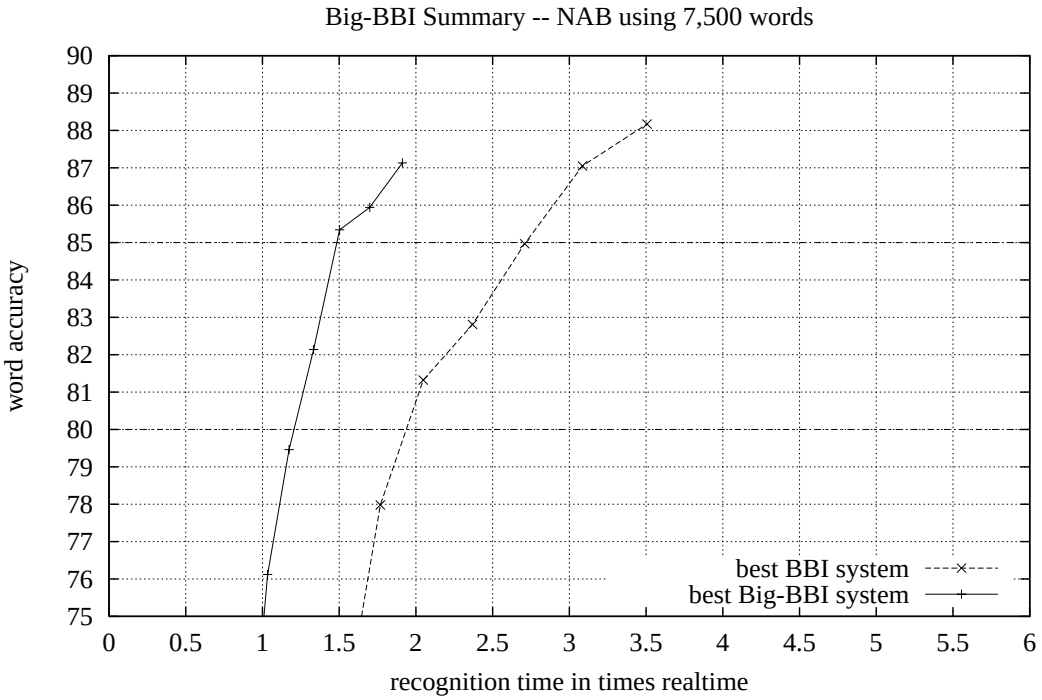


Figure 5.38: Generalized-BBI versus BBI for NAB with 7,500 words

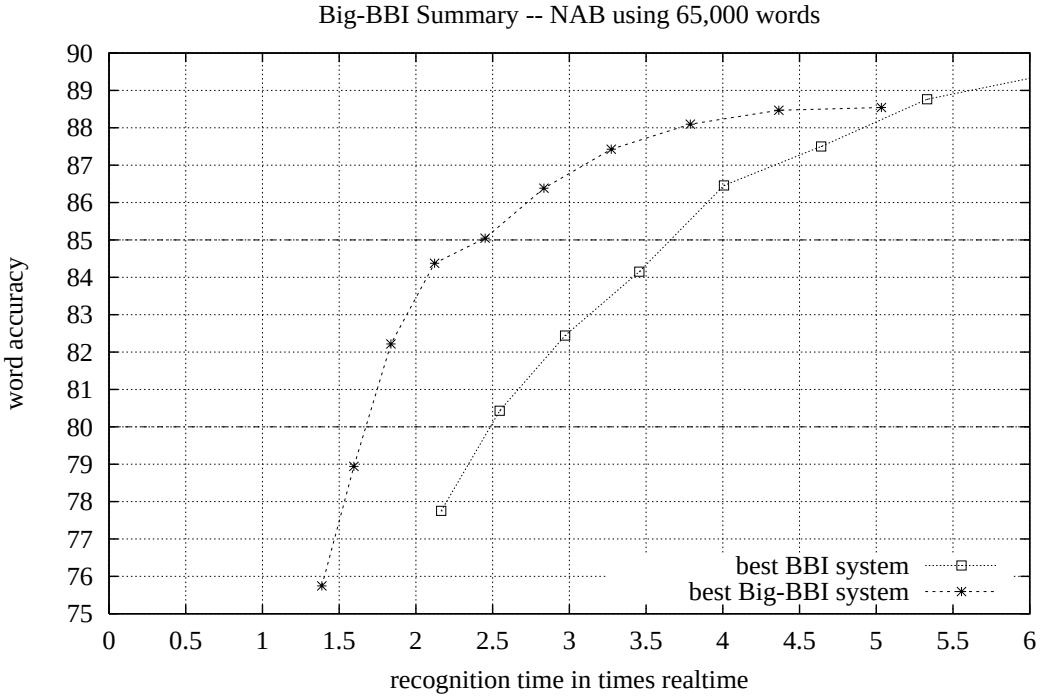


Figure 5.39: Generalized-BBI versus BBI for NAB with 65,000 words

5.3 Skipping Input Frames

Even for systems that use the Generalized-BBI algorithm, more than half of the total CPU-time is still used for score computation in the case of the 7,500 word recognizer, and more than 40% in the case of the 65,000 word recognizer.

The original intent of the experiments in this section was to reduce the number of score calls by skipping the computations for some input frames. However, as the second part of the section shows, is more efficient to skip whole input frames rather than only score computations.

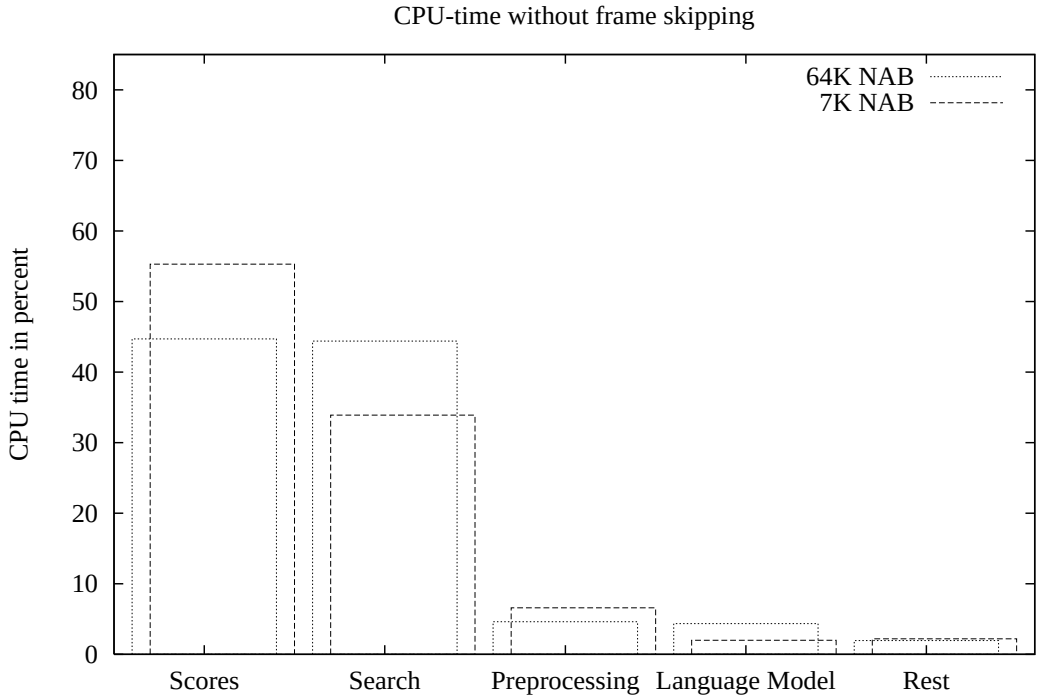


Figure 5.40: Runtime Profile using Generalized-BBI

5.3.1 Skipping Score Computations

Since at a frame rate of 100 frames per second speech is a slowly changing signal, the observation probabilities do not usually change dramatically from one frame to the next. Also, there is no faster way to compute a score than to assume it has not changed, and not compute it at all. If high recognition speeds are required, the error introduced by this approximation can be recovered by using wider beams. For the experiment in this section, the score computations were skipped every other frame, independent of the dynamic behavior of the input signal.

Figure 5.41 shows that the speed-up for the 7,500 word vocabulary is around 20%. The maximum word accuracy at large beams is reduced to below 88%.

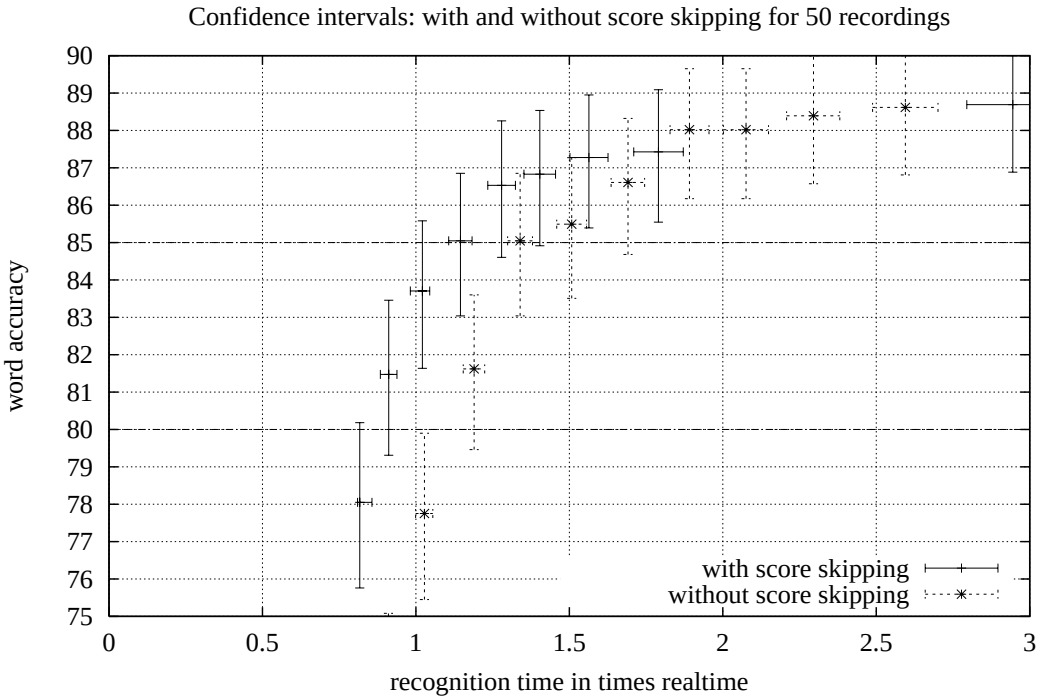


Figure 5.41: Skipping score computations, 50 recordings with 95% confidence intervals.

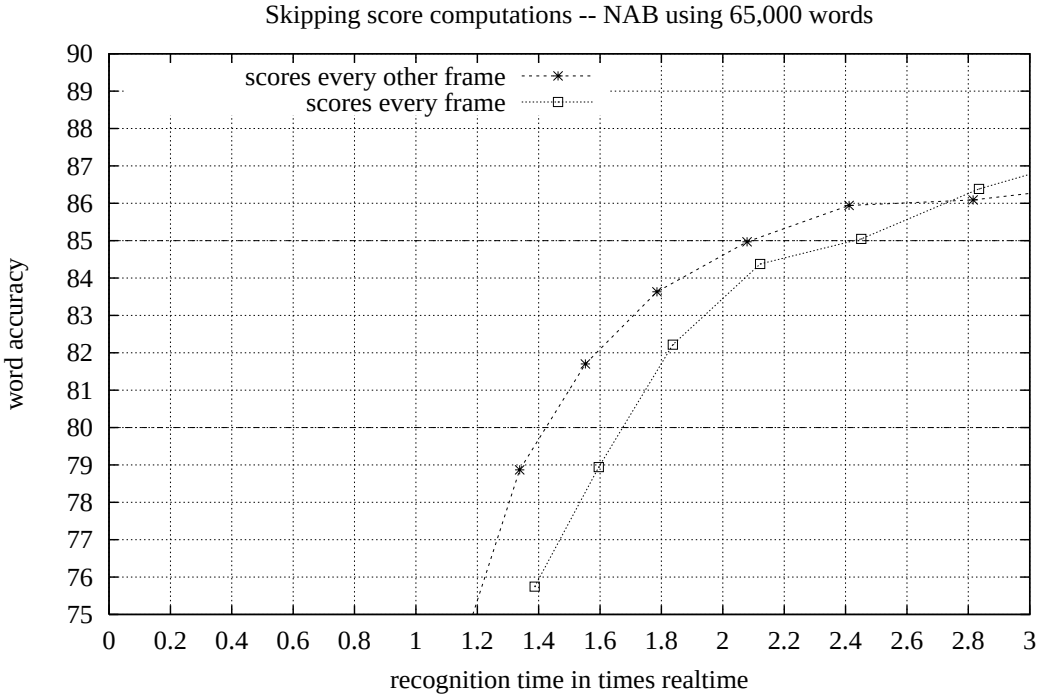


Figure 5.42: Skipping score computations for NAB with 65,000 words

5.3.2 Reducing the Frame Rate

If computing the scores only every other frame still produces a reasonable recognition output, it is worth trying to completely skip every other frame during the preprocessing. The following issues need to be considered:

- To avoid retraining, the input vectors must still match the Gaussians in the codebooks. Smoothing or deviations in delta-computations of the preprocessing should be avoided.
- The topology of the phoneme-models should be extended by allowing longer skip transitions to compensate for the compressed time-axis in the *Viterbi* algorithm.
- While the number of acoustic scores per sentence is cut in half, the number of word transitions remains the same. Instead of adjusting the language model parameters, it is possible to scale the acoustic scores to compensate for this.
- Lookahead weights and other frame-rate related parameters have to be adjusted.

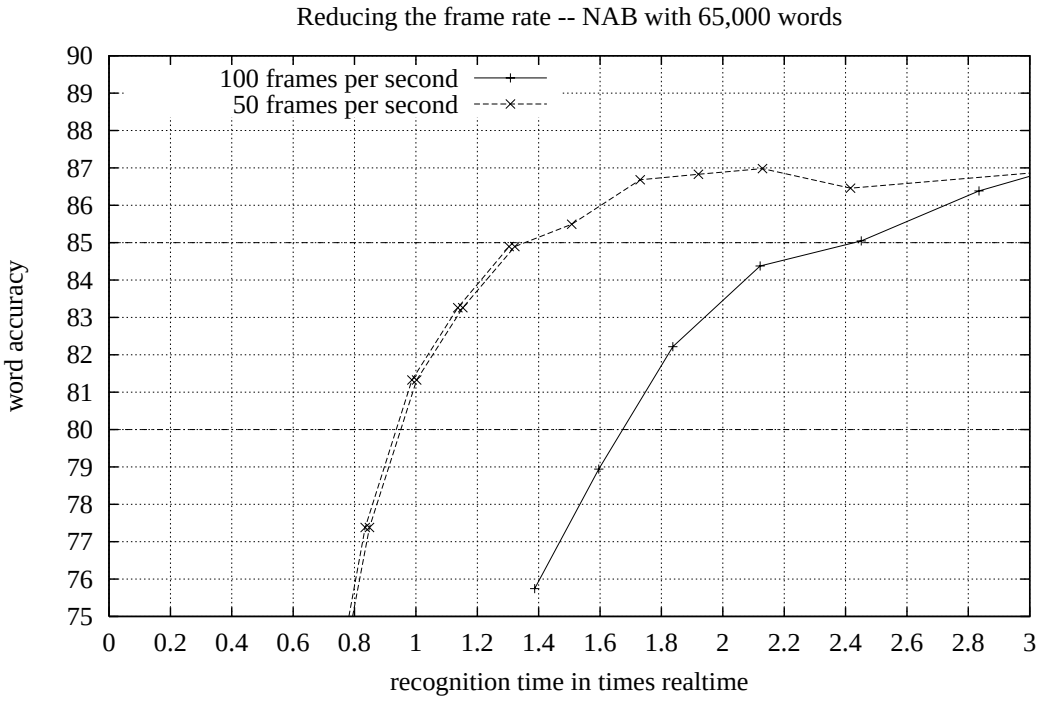


Figure 5.43: 50 versus 100 frames per second. Parts of the plots for 50 fps have been recomputed using a slightly different hardware configuration to verify the results. However, the deviations are very small.

Figure 5.43 shows that reducing the number of frames per second yields a considerable speed-up. What is not shown in this plot is that the maximum achievable word accuracy drops to 87%. This method should therefore only be used if very high recognition speeds are required.

However, using the frame skipping, a recognition with a vocabulary of 65,000 words is now possible in real time, at an error rate of only 18.5%.

5.3.3 Conditional Frame Skipping

One of the drawbacks of the frame skipping technique described earlier in this section is that for rapid speech the error rate increases can be prohibitive, especially if the concept is extended from skipping only every other frame to skipping up to two out of three frames.

Two considerations go into the estimation of the number of frames to be skipped:

- If the signal remains static for the next couple of frames, the exact behavior for the next frame compared to the current frame may be less important than if the signal is about to change.
- If the recognition process is expensive at the current frame, that is if a large number of states is active, skipping the next frame yields a larger speed-up compared to simple sections of the speech signal, where only few states are active.

To estimate whether the signal is static with respect to the acoustic models, the output of the HMMs for the lookaheads can be used: Computing the observation probabilities $f(x|s)$ for these models is fast enough to be performed for every input frame x and every state s of the lookahead HMM model. The Euclidean distance between the vectors of these scores for the next two frames is used to determine whether or not the signal is currently changing:

$$D(x) = \sum_{s=0}^{S} (f(x+1|s) - f(x|s))^2 \tag{5.3}$$

To normalize this value, the current distance is divided by the maximum distance observed up to the current frame in the test set:

$$D_{norm}(x) = \frac{D(x)}{\max_{0 \le i \le x} D(i)} \tag{5.4}$$

For the presented experiment on conditional frame skipping, the normalized score vector distance was used as follows:

$$\begin{aligned}
0.0 < D_{norm}(x) \le 0.3: &\quad x_{next} = x + 3; \quad \text{(skip two frames)} \\
0.3 < D_{norm}(x) \le 0.6: &\quad x_{next} = x + 2; \quad \text{(skip one frame)} \\
0.6 < D_{norm}(x) \le 1.0: &\quad x_{next} = x + 1; \quad \text{(skip no frames)}
\end{aligned} \tag{5.5}$$

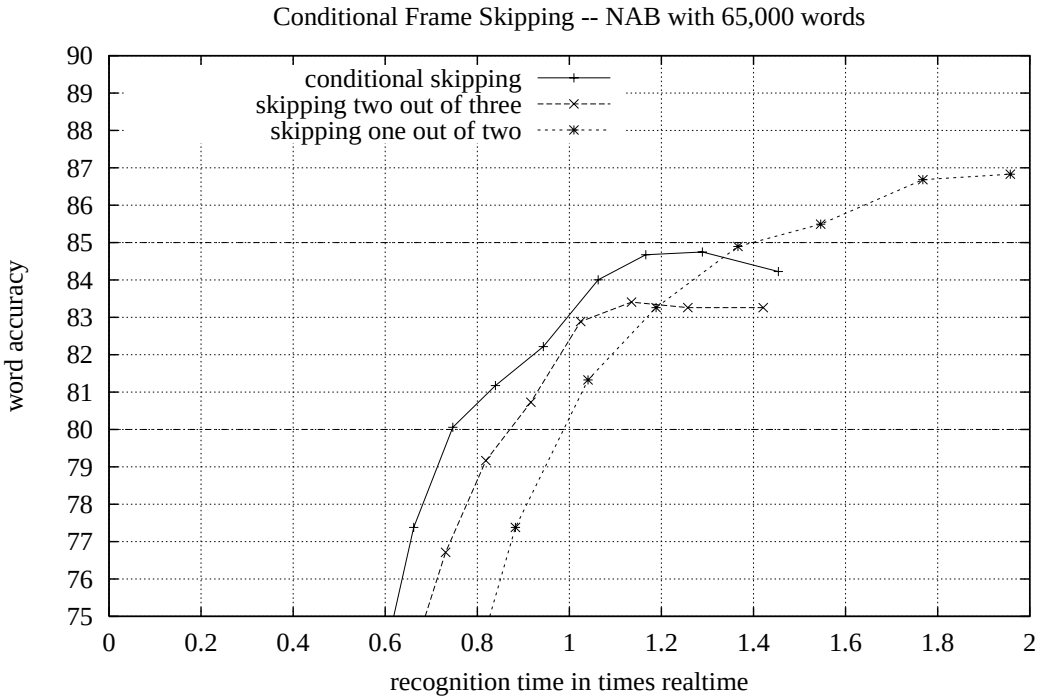


Figure 5.44: Results for the conditional frame skipping on NAB using a 65,000 word vocabulary. Unlike in the previous experiments on frame skipping, the lookahead scores are now computed for every frame.

Figure 5.44 shows the results for this experiment: while skipping two out of tree frames yields a faster recognition than skipping one out of two frames, the word accuracy is also smaller. Using the conditional frame skipping, the loss is considerably smaller. Note that the conditional frame skipping only appears to be faster than always skipping two out of three frames: if the points corresponding to the same beam setting are compared, the conditional skipping is a little slower. However, the corresponding word accuracy is up to 4% better when compared to the simpler approach.

Chapter 6

Application Oriented Issues

This chapter gives additional information about some specific application oriented issues of the algorithms introduced in this thesis. Since most experiments during the development of the algorithms were performed off-line and from pre-recorded data, they do not capture possible effects to systems with actual users and limited per sentence response time.

The first section illustrates how the algorithms presented in this thesis performed when applied to new tasks. As a first portability test, all methods were tried on the subset for male speakers, mostly to check against possible tuning to the test set. For further portability studies, the system used for the *German Spontaneous Scheduling Task* (GSST) is presented in more detail. → 101

The second section addresses the issue whether robustness of the recognizer towards noise and foreign accents suffered by the introduction of the new algorithms. However, since no effort had previously been made to increase robustness for the particular setup used as baseline in this thesis, many results are not conclusive and may not generalize to other setups. To reduce the lack of robustness in the baseline recognizer would have been beyond the scope of this thesis. → 105

The third section explains how a system using all of the new algorithms can still be fully pipelined, allowing recognition output before the end of the utterance has been detected. → 107

Finally, an simple application demonstrator that successfully integrates all of the above techniques is described. → 108

6.1 Portability

Many of the presented algorithms that have been developed on the subset for female speakers of the NAB dictation task been ported to a variety of different tasks. As a first portability test, all methods were tried on the subset for male speakers, mostly to check against possible tuning to the test set.

The Generalized-BBI algorithm, the unigram and phoneme lookaheads, and the general knowledge about pruning behavior were also successfully applied to build faster recognizers for the German, English, and Japanese spontaneous scheduling tasks as part of the *VerbMobil* project, and for a Japanese dictation system.

The system used for the *German Spontaneous Scheduling Task* (GSST) is presented in more detail to show the algorithms ported to new tasks.

NAB for male speakers

But for a few exceptions, the experiments in chapters 4 and 5 have been performed on a subset of the NAB evaluation data that contains only female speakers. Making so many decisions on a limited set of data quickly leads to algorithms that will not work well for other data. To be certain this did not happen here, the most important techniques, like the setting of the pruning thresholds, the unigram and phoneme lookaheads, the use of the Generalized-BBI algorithm with the *Nearest Neighbor Approximation* and the skipping of score computations[1] were reused for the male system without adapting or tuning any parameters. The only difference when compared to the system used on the sentences by female speakers was the use of the appropriate codebooks and weights for the gender dependent recognizer. Also, a new Generalized-BBI tree had to be computed for use with the male codebooks. The lookahead system is gender independent and was reused without any change.

Figures 6.1 and 6.2 show that porting the algorithms that have been developed and tested for the female subset of the NAB data to the male subset yield satisfactory results without adjusting any parameters.

German Spontaneous Scheduling Task (GSST)

The spontaneous speech used for this test contains transcribed human-to-human dialogues, mostly taken from the *VerbMobil* project.

The acoustic variability in the spontaneous data is higher, making the recognition task harder. However, since the scenario is limited to scheduling dialogues only, the task is covered well by the statistical language model and the out of vocabulary rate is low for a vocabulary of only 5,500 words.

All techniques used for the NAB system were successfully ported to GSST, and some of them were used for the integration of the 1997 VerbMobil prototype.

This faster system was based on the recognizer used for the 1996 VerbMobil evaluation. The gender independent evaluation recognizer had 2500 codebooks, each with 32 Gaussians in 32 dimensional space. Over those codebooks, 10,000 weights were used to model further allophones. To reduce the speaker to speaker variations, a

[1] Skipping whole frames was not tried until after the experiments for this section.

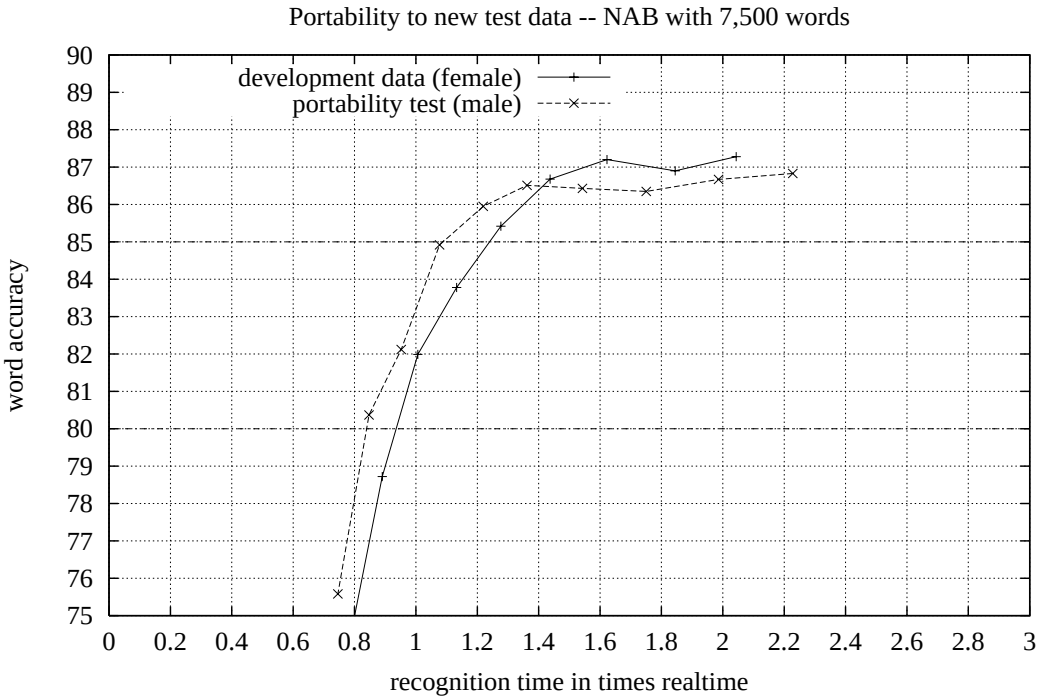


Figure 6.1: The male and female NAB systems for a 7,500 word vocabulary behave very similarly.

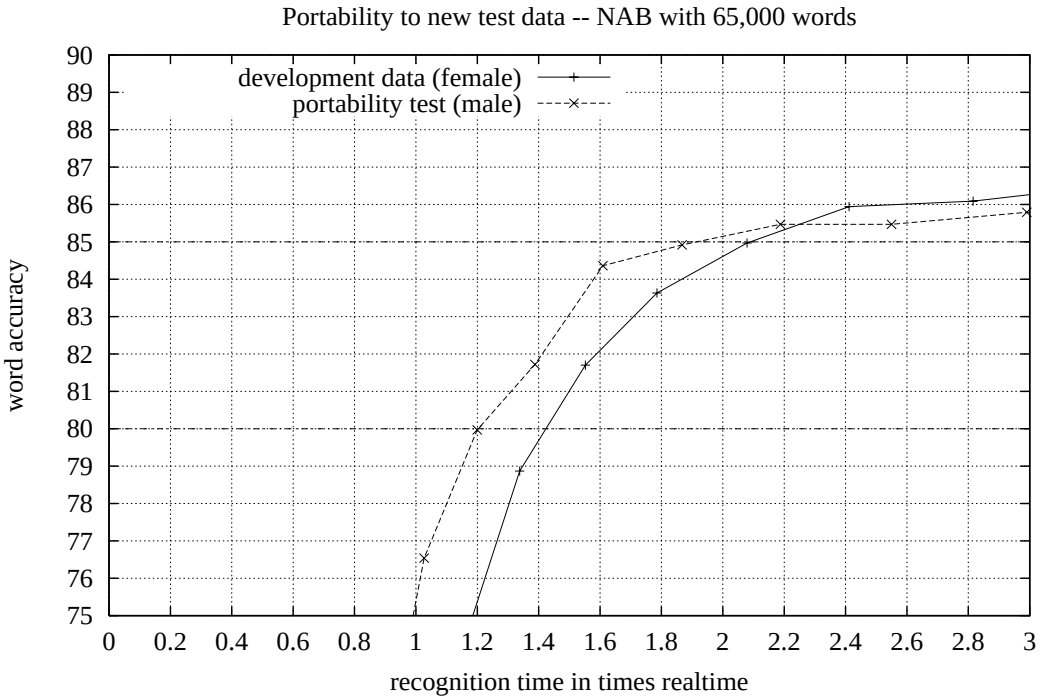


Figure 6.2: Using the vocabulary size of 65,000 words the male system (unseen data) and the female system used to develop the algorithms still behave the same.

vocal tract length normalization (VTLN) was used. The required warping factor for this algorithm was determined using the output of a preliminary recognition run.

The total recognition process for one sentence was divided into 7 to 8 steps:

1. Tree search using a neutral VTLN parameter
2. Linear search using a neutral VTLN parameter
3. Re-scoring the word graph using a trigram language model
4. Forced Alignment of the resulting hypothesis to determine a better VTLN parameter
5. Tree search using the new VTLN parameter
6. Linear search using the new VTLN parameter
7. Re-scoring the new word graph using a trigram language model
8. Optional speaker and channel adaptation

	RT	WA
Evaluation system without adaptation or adjudication	200	84.9
Using 2500 instead of 10,000 weight sets	70.5	84.9
Nearest Neighbor Approximation	32.0	84.4
Generalized-BBI	17.0	83.6
VTLN between tree and linear search only	10.3	83.1
3-state phoneme model instead of 6-state	8.2	83.1
Phoneme *lookaheads*	7.3	82.8
Retraining without any VTLN	5.3	81.9
Generalized-BBI for lookaheads, improved pruning	4.3	81.5
Tree search only, no linear search	2.3	80.0

Table 6.1: Speeding up the *VerbMobil*-recognizer for the 1997 integration.

Figure 6.3 compares the Generalized-BBI system used for integration into the 1997 prototype to several alternative algorithms to compute the observation probabilities. The Generalized-BBI and BBI trees were chosen after some testing on a cross-validation set and have a depth of 4 for the 2,500 BBI trees and a depth of 8 for the one Generalized-BBI tree.

Also shown is the pruning behavior for the Generalized-BBI system with a frame rate of 50 instead of 100 frames per second. This system was not used for the integration because it does not reach the required minimum word accuracy of 80%. Reducing the global frame rate on a recognizer for spontaneous speech is dangerous because of frequent sections with rapid speech.

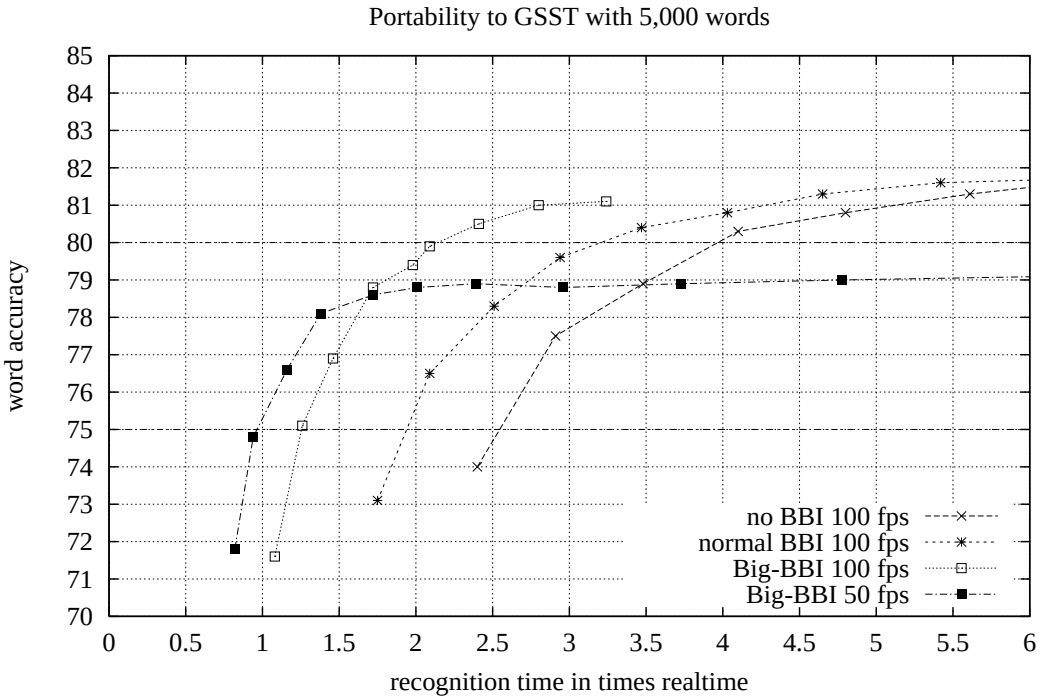


Figure 6.3: The integrated system using Generalized-BBI in comparison to 1) the same system with frame skipping and 2) to a system with normal BBI only and finally 3) to a system using only the *Nearest Neighbor Approximation*.

The requirements for the integrated system, namely to achieve a minimum word accuracy of 80% at a recognition time of less than 3 RT was met by porting the algorithms developed in this thesis to the new task.

6.2 Robustness

Usually, the robustness of a recognizer is reduced if small pruning thresholds are used. If the speech signal used to test the recognizer deviates from the training conditions by new background noises, dialects or foreign accents, or by using a different microphone, the word accuracy for a recognizer with tight pruning thresholds will suffer more than the accuracy of a system with large beams.

This is not necessarily the case for the other methods presented here to reduce the recognition time. *Phoneme lookaheads* use small acoustic models, that are less likely to be over-specialized than the big models of the main acoustic and might even improve the recognition results.

To find out to what degree the recognizers robustness was affected by increasing the recognition speed, a number of experiments has been performed on data that deviates significantly from the training conditions of the recognizer. Using the evaluation data of the *spoke conditions* for the 1994 NAB evaluation, tests were run on

speech with foreign accents (S3) and low quality microphones (S5).

The official language model for this test would have used a closed 5,500 word vocabulary. However, as the goal of these experiments is to compare the recognition under adverse conditions to the clean speech results presented in earlier sections, they were run reusing the same language model and vocabulary as the clean speech experiments. The resulting vocabulary size and perplexity is larger compared to the official evaluation conditions. The presented error rates should not be compared to those published in [ARPA Workshop, 1995].

For the s3-tests on foreign accent speech, 111 sentences were chosen randomly from the original test set so that each of the 10 speakers is equally represented. The native languages for the 10 speakers are British English, Mandarin Chinese, Norwegian, German, Japanese, Indian, Croatian, Mexican Spanish, Hebrew and French.

For the S5-tests on different microphones, 127 recordings were chosen randomly from the original test set, with an equal representation of all 20 speakers and all 10 microphones. Two of the 10 microphones are dynamic microphones, the remaining eight use a condenser system. No headset mounted microphone was used in the recordings of this test set. All training data used for the NAB recognizer was recorded using a dynamic headset mounted microphone (Sennheiser HMD 410).

Test set	Subset	WE baseline		WE fast recognizer	
NAB-94 normal	female	8.9 %		12.7 %	
NAB-94 normal	male	9.2 %		13.3 %	
NAB-94 accent	female	34.4 %	(*3.9)	47.7 %	(*3.8)
NAB-94 accent	male	21.9 %	(*2.4)	30.3 %	(*2.3)
NAB-94 micro.	female	15.0 %	(*1.7)	26.7 %	(*2.1)
NAB-94 micro.	male	21.1 %	(*2.3)	35.2 %	(*2.6)

Table 6.2: Robustness of the fast versus the baseline recognizer. The factor given in parentheses specifies the error rate increase over the clean speech experiment. Since no measures were taken to improve the robustness of the baseline system, the significance of these results is very low.

Table 6.2 lists the results for the NAB baseline system and the fast recognizer on a 7,500 word vocabulary. The error rates for the fast recognizer were taken at two times real time, assuming that for adverse conditions run times slower than real-time are acceptable. This system is still using tight beams that lead to a slight performance loss for bad acoustics.

The error rates for the baseline system were obtained using the settings that yield the best results for clean speech. They still include a phoneme lookahead which does not affect the word accuracy at the conservative beams settings used. The recognition times for this system are at about 90 to 100 times real-time (200 RT without lookaheads).

Since the baseline system has been tuned to perform well on the very clean data of the main evaluation test set, and since no attempt was made to increase the robustness of the system, the loss of performance for adverse conditions is dramatic but not surprising. Unfortunately, this also considerably reduces the significance of the experimental results presented in this chapter. Adding preemphasis to the preprocessing and using *Mel-cepstrum* coefficients rather than *Mel-scale* coefficients, speaker adaptation, and adding VTLN as well as noisy training data to train a smaller system, all of these are only a few of the many steps that need to be taken to solve this problem. As this experiment was about comparing the fast to the baseline recognizer, and not on how to improve the baseline robustness, these steps would have gone far beyond the topic of this thesis.

However, recent results [Kubala et al., 1997c] on the 1996 DARPA Hub-4 Evaluation → 116 on broadcast news show that the possibilities for improvements on speech of foreign speakers are somewhat limited: while the reported recognition rates on clean speech by American speakers are 14.9% without and 13.5% using speaker adaptation, the corresponding values for foreign speakers are 26.4% without and 23.2% with speaker adaptation.

6.3 Pipelining

While the speaker is still speaking, the recognizer can perform useful tasks such as preprocessing, computing observation probabilities, and search. The earlier the user gets feedback from the system, the better he can judge whether the recognizer is performing as expected. If the system has been designed to allow pipelining, and if the recognizer performs well enough under real-time constraints, the response time of the resulting system depends only on the small, constant delays introduced during the pipelining.

If any recognition output is to be produced before the speech input is over, algorithms that require all of the utterance can not be used in the process. One algorithm that has to be modified for pipelining is the normalization in the preprocessing, which uses the value of the highest amplitude in the utterance or a mean over all frames in the utterance. Long range language models and forward backward search strategies can also cause problems.

Such maximum and mean values have to be computed as running means over a window that only requires a small number of frames beyond the point for which the preprocessing output is computed. If the search strategy disagrees with a run-on implementation, it is often possible to create a preliminary output with a search that can be pipelined, and then update that output with all available information at the end of the utterance.

However, if the system has not been planned and trained to run in a pipelined fashion, the approximations introduced to this end can reduce the performance of the system in terms of speed and word accuracy.

The only restrictions that the fast NAB system imposes on the pipelining of the recognition process are as follows

- The *phoneme lookaheads* need up to five frames of speech beyond the point examined by the search. These five frames correspond to less than a tenth of a second of speech data. Therefore, any delay caused by the lookaheads will be on the order of a tenth of a second, which is insignificant for most applications.
- For the current preprocessing, several steps use a normalization over the whole utterance. Those have to be replaced by a normalization over a decaying window. Without retraining of the recognizer, the resulting deviation from the training conditions leads to a loss in word accuracy of about 2% absolute.

Since the NAB-System loses more than 5% word accuracy absolute due to pruning in the area between one and two times real time, the use of pipelining for an online system pays off: if the time that the user has to wait for the recognition output is kept the same, the recognizer can spend twice as long to process the input because it can do useful work during the recording time. The resulting increase in word-accuracy of about 5% more than compensates the loss due to normalizing problems.

Note that no matter how the recognizer has been trained, the pipelining only leads to a subjective speed-up. The actual number of cycles used for the recognition is always larger in pipelined mode since the search structure becomes more complicated (e.g. due to frequent backtraces).

6.4 The NAB Dictation System

To demonstrate how prototype applications can be built using the results from the work performed in this thesis, this section describes the English dictation system built based on the JRTk Toolkit. Further details on the scenario and recognizer are
→ 31 found in chapter 3.

The goal of the dictation system is to provide fast speaker independent recognition of English speech, using a vocabulary size of 20,000 words or more. The word accuracy depends on the recognition speed, and can be adjusted using a slider which influences the beam settings. For real time performance, the average initial word accuracy for native speakers is about 80%. The system can be adapted to a new speaker using a learn button, which invokes a maximum likelihood adaptation step on the recognized data so far.

Most of the recognition happens while the speaker is still speaking, minimizing the actual time the user has to wait for the system. The confidence in the recognition output is computed by analyzing the variability of the word-graph and represented with the gray level used to display the text.

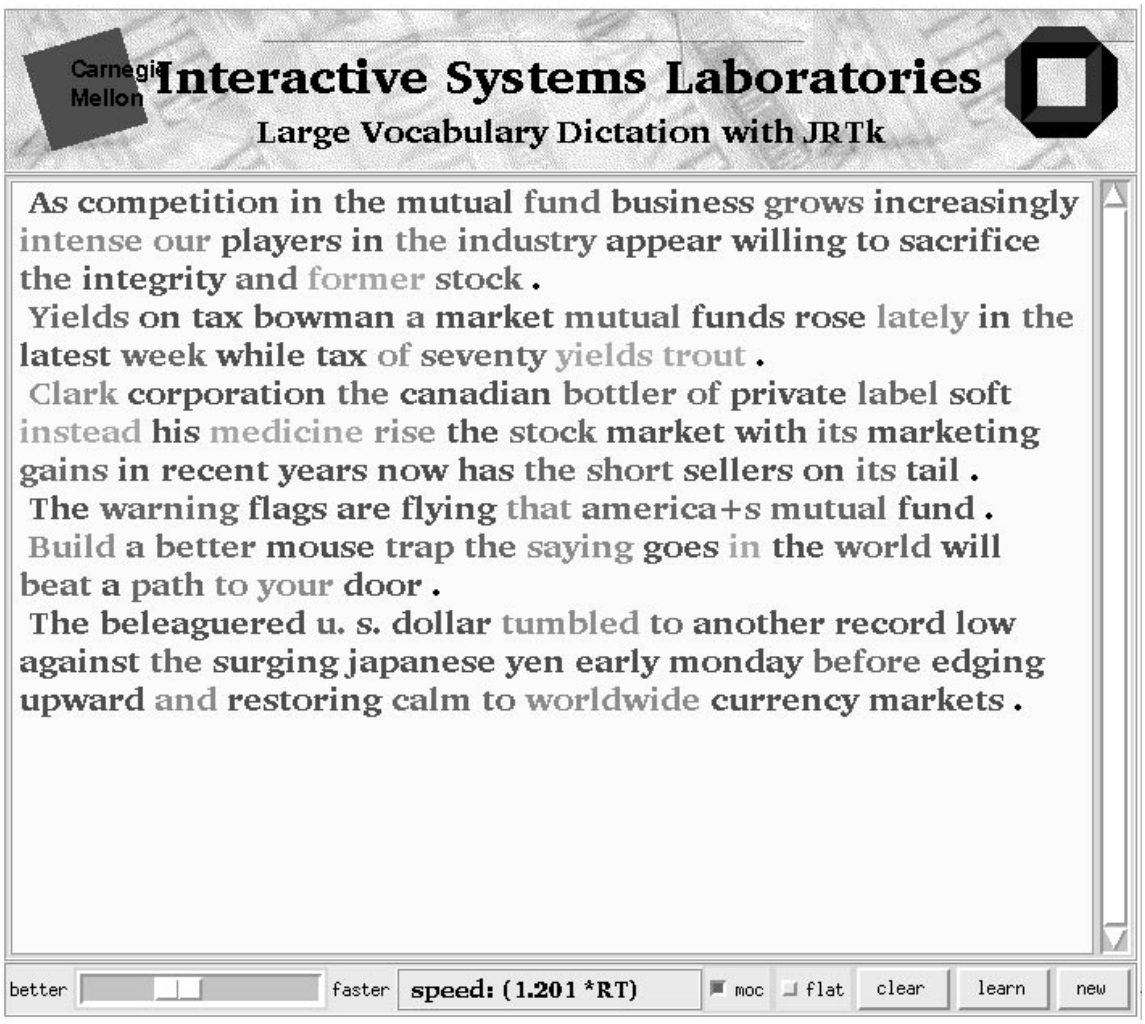


Figure 6.4: Example output of the recognition system. The actual input was taken from the test set: *As competition in the mutual fund business grows increasingly intense, more players in the industry appear willing to sacrifice integrity in the name of performance. Yields on taxable money market mutual funds rose slightly in the latest week, while tax exempt yields dropped. Cott corporation, the Canadian bottler of private label soft drinks that has mesmerized the stock market with its marketing gains in recent years now has the short sellers on its tail. The warning flags are flying at America's mutual fund. Build a better mouse-trap the saying goes and the world will beat a path to your door. The beleaguered US dollar tumbled to another record low against the surging Japanese Yen early Monday before edging upward and restoring calm to worldwide currency markets.* The corresponding word accuracy is around 80%.

To help correct remaining recognition errors, words and sections of the recognized text can be replayed by clicking on them.

Remaining problems are the high memory usage (more than 300 MBytes of RAM), and the low robustness with respect to changing recording conditions and foreign accents. Also, the usability of the statistical language model that was built using data collected in 1994 and 1995 is low for new data. It has to be updated regularly

using commercially available databases. If a dictation system is used frequently, rebuilding the language model with the newly dictated speech is also an option. Finally, as can be seen in figure 6.4, the word accuracy of 80% may not be enough for many applications.

Chapter 7

Conclusion

When you make your mark in the world,
watch out for guys with erasers.
– The Wall Street Journal

7.1 Summary

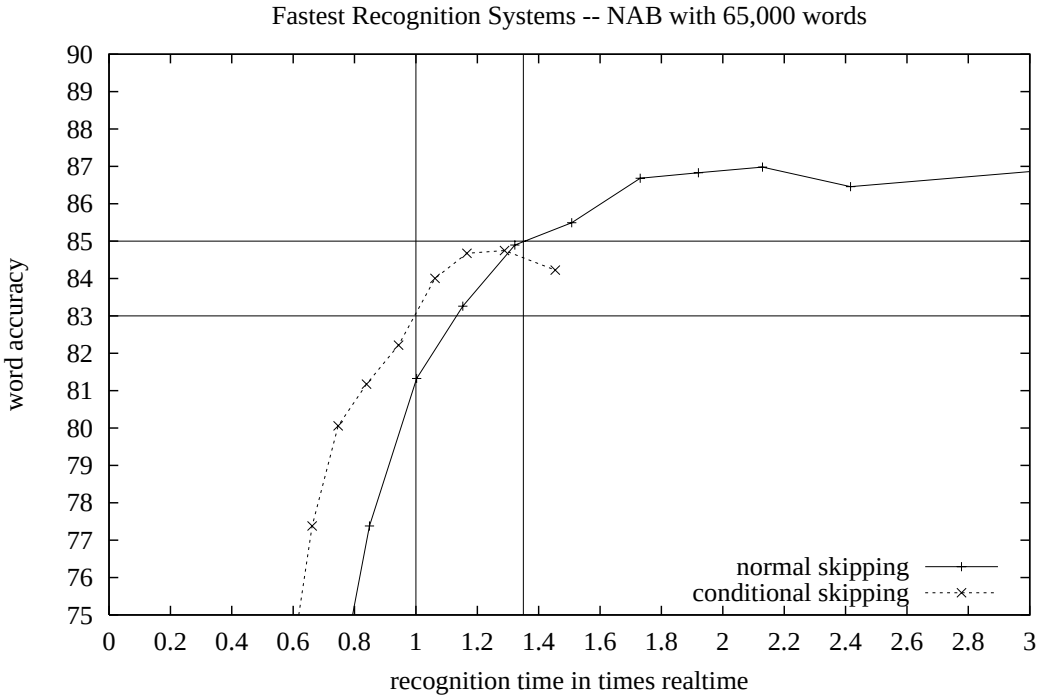


Figure 7.1: Pruning behavior of the currently fastest NAB system on a vocabulary with 65,000 words.

Using the techniques that have been developed as spin-offs from the research presented in this thesis it is now possible to recognize speaker independent, continuously spoken speech in real time with JRTk, using a vocabulary size of 65,000 words.

The corresponding error rate on the NAB task is 17%; if an error rate of below 15% is required, this can be achieved at only 1.35 times real time. Figure 7.1 shows the pruning behavior of the so far fastest NAB system.

The total speed-up is composed of many contributions, including:

- Demonstrating that the loss of 0.1% to 0.3% absolute in Word Accuracy due to the introduction of the *Nearest Neighbor Approximation* is small compared to the possible speed-up of more than a factor of two. → 45
- Evaluating when to use only the first pass of the new two-pass search and when to use both passes. → 56
- Comparing and combining different pruning strategies. → 61
- Introducing *Unigram-Lookaheads* that make a tree-search viable for vocabularies between 2,000 and 65,000 words. → 69
- Introducing *Phoneme-Lookaheads* with a new approach based on context-independent HMMs that fits well into the framework of an HMM-based recognizer. The resulting speed-up of this is again about a factor of two. → 72
- Developing the *Generalized-BBI*, a generalized version of the *bucket-box-intersection* algorithm that works well for fully continuous HMMs, and combining it with the *Nearest Neighbor Approximation*. The total speed-up over just using the *Nearest Neighbor Approximation* is again about a factor of two. → 87
- Using a data driven conditional frame-skipping approach to speed up the recognition process for stationary portions of the speech signal. → 99

Also, low level optimizations and insights gained on approaches that either did not work, or did not combine well with algorithms helped to achieve the goal of a real time recognition system on a single processor of a Sun-Ultra workstation with a 167 MHz clock-rate.

7.2 Assessment

A number of companies are pushing into the large vocabulary speech market with considerable effort. For a marketable software, recognition speed is as important as accuracy. Recently, some companies have developed LVCSR systems that provide high accuracy at near real time performance. However, these systems differ in significant aspects from the JRTk system.

Recently Dragon Systems started marketing their 30,000 word speaker dependent continuous speech recognition system *Dragon Naturally Speaking*, that runs on a 133MHz Pentium PC. This product is the result of several years of research on fast LVCSR. The main two differences when compared to the JRTk are that the dictation system is tuned to be small and fast but not as flexible as a toolkit, and that it is speaker dependent. Using speaker dependent models allows the use of much smaller acoustic models, avoiding the problem of the high costs for score computation, which was one of the main points addressed in this thesis. Speaker dependent models also tend to discriminate better, allowing a more aggressive restriction of the search space. In November 1997 IBM released its *ViaVoice Gold* product, which supports a similar functionality. To get the full dictation functionality, the user is required to read 256 enrollment sentences that are used to adapt the system to the user. → 122

A comparable system for Pentium-PC's proposed by Siemens Corp [Niemöller et al., 1997] is speaker independent with a vocabulary size of 10,000 words. However, to achieve real time performance on a PC platform, special-purpose hardware is used for the observation probabilities, leaving only the search to the main CPU. → 126

The most recent results for NAB, with a vocabulary size of 20,000 words, from the research group of Hermann Ney, usually regarded as the reference site for fast speech recognition in Europe [Ortmanns et al., 1997a, Ortmanns et al., 1997b], are around 8 to 16 times real time for error rates over 16%. → 125

The fast JRTk resulting from the work presented in this thesis therefore compares favorably to the systems presented by the international community. An important benefit of the fast JRTk System is that while it is as fast and accurate as the competing dictation systems, it is nevertheless a flexible research system.

7.3 Perspectives

The fast recognition systems based on JRTk are derived directly from full evaluation systems. Therefore, the reduced recognition times also helps to reduce the cycle time for the development of research systems, hopefully leading to recognizers that provide even higher accuracies at high recognition speed.

On the long run, some of the algorithms presented in this thesis may become obsolete: for instance, it is not yet clear whether mixtures of Gaussians or Neural Nets are the better approach to estimate observation probabilities.

However, the current impression is that the combination of mixtures of Gaussians and Neural nets may yield the lowest error rates, resulting in a continuing demand for algorithms like the Generalized-BBI.

Unless a dramatic breakthrough in pattern matching introduces a completely new paradigm for the search problem, the tree search and lookahead algorithms will remain useful for a long time.

Finally, using a mode dynamic view of the input stream, allowing for changes in frame rate and maybe even the selection of features within one utterance, may become an important step towards faster systems and higher accuracies.

Appendix A

Other LVCSR Systems

This appendix gives an overview of large vocabulary continuous speech recognition systems other than the JRTk in order to define the global context in which this thesis took place. It also provides some comprehensive information about systems that have been referred to elsewhere in this thesis.

Many of the described systems can be configured to use a variety of vocabularies and acoustic and language models. Since the test-set of the 1994 NAB evaluation served as baseline for most experiments in this thesis, the setup reported for this evaluation is given where available.

To show the progress in some systems, and to give more detailed information about systems that did not participate in the 1994 NAB evaluation, the system configurations for the 1996 Broadcast News Evaluation is also provided for some participants.

Since the number of research groups working on speech recognition is very large, the following sections cannot describe all existing systems. However, the author has attempted to select systems and setups that are representative for the ongoing work in the speech community.

A.1 AT&T

The AT&T recognizer is an HMM based system that uses a standard hidden Markov model approach for acoustic modelling with triphones. It features a unique bottom up search strategy that represents one of its key differences to the JRTk system used in this thesis. Also, for fast estimation of the observation probabilities AT&Ts recent systems use Gaussian Selection techniques rather than the binary space partitioning approaches pursued in this thesis.

Other recent work in the AT&T LVCSR research includes the development of parallel speech recognition algorithms and the automatic determination of parameters.

November 1994 NAB Hub1 Evaluation

For this evaluation, 39 features were computed in the preprocessing step, consisting of 12 cepstral parameters, the log energy, and their first and second derivatives.

The system used gender dependent triphone models. Where the training data was insufficient to estimate accurate triphone models, a back-off to left-context or right-context diphone models or context independent models was used. The resulting system had 12,500 codebooks, each containing between four and 15 Gaussians with diagonal covariances. Over these codebooks, additional distribution weights defined a total of 22,000 context dependent phoneme models.

To find the best recognition hypotheses the system used two main passes, the first to build a word graph, the second to re-score the word graph based on full context dependent acoustic models and 5-gram language models. The first pass was subdivided into several steps, including phoneme recognition, syllable recognition, word recognition and bigram application. Each of these step produces a lattice, that was used to restrict the search space for the subsequent step.

With this setup, the AT&T system reached a 13.0% word error rate for the NAB-C1 (contrast) evaluation conditions.

Selection of Papers

For further information about the AT&T system, refer to [Ljolje et al., 1995], [Mohri and Riley, 1997], [Phillis and Rogers, 1997], and [Bocchieri and Mak, 1997].

A.2 BBN – Byblos

The BBN Byblos recognizer is an HMM based system that uses a multi-pass decoder based on the forward-backward search developed by BBN. More expensive and accurate acoustic models were used in successive passes. Neural Nets for acoustic modelling were applied to re-score N-Best lists. The multi-pass forward-backward search that uses complex acoustic models only after the search space has been considerably reduced by earlier passes eliminates the need for faster computation of observation probabilities. However, this approach is difficult to use for real-time systems since the result of the forward pass is unreliable, and the backward pass cannot start until after the end of the utterance has been detected. Recent LVCSR research includes work on efficient decoders and multi-lingual speech recognition.

November 1994 NAB Hub1 Evaluation

For this evaluation, a preprocessing step with 45 features, consisting of 14 cepstral coefficients energy, and their first and second derivatives was chosen.

Two acoustic models with different complexity were used, both based on 5 state hidden Markov models. The less complex model consisted of 46 context independent codebooks, each with 64 Gaussians and diagonal covariance matrixes. Over these codebooks, distribution weights defined 30,000 allophones. This model is referred to as phonetically tied mixture model (PTM).

The more complex model had 5,000 context dependent codebooks with 32 Gaussians each, again with diagonal covariances. Over these 5,000 codebooks, distribution weights again defined 30,000 allophones. This model is referred to as state clustered tied mixture system (SCTM).

To find the best recognition hypotheses, BBNs developed its own multi-pass forward-backward search algorithm:

1. Fast match in forward direction produced word ending scores using a simplified time synchronous search.
2. Time synchronous backward pass using within word PTMs and bigrams produces word starting points and scores.
3. Forward oriented time synchronous search using the same models as the previous pass provides updated word ending scores.
4. The resulting lattice was processed using cross-word SCTM models and trigrams to produce an N-best list.
5. The N-best list was re-scored with neural networks.

With this setup the Byblos system got an 11.9% word error rate for the NAB-C1 (contrast) evaluation conditions.

November 1996 broadcast-news evaluation

For this evaluation, BBN used a modified multi-pass recognizer:

1. The first pass was a time synchronous forward search with bigrams and within word triphones (45 codebooks with 256 Gaussians each) which created a lattice of word end times and scores.
2. The subsequent search pass built an N-best-list using approximate trigrams and within-word SCTM models (1,000 codebooks with 64 Gaussians each).
3. Finally, a re-scoring step with cross-word triphones and trigrams was used to produce the 1-best recognition.

For the November 1996 Broadcast News HUB-4 evaluation the system used a 45,000 word vocabulary and achieved a 30.2% word error rate.

Selection of Papers

For more information about the BBN system, refer to:
[Austin et al., 1991], [Schwartz and Austin, 1991], [Schwartz et al., 1992],
[Austin et al., 1992], [Nguyen et al., 1994], [Nguyen et al., 1995],
[Nguyen and Schwartz, 1997b], [Nguyen and Schwartz, 1997a],
[Kubala et al., 1997a], [Kubala et al., 1997b], and [Kubala et al., 1997c].

A.3 Cambridge University – ABBOT

The ABBOT system differs significantly from JRTk and all other systems described in this appendix by being the only neural net driven system used in large scale evaluations. It is also among the few systems to use a stack decoder rather than a time synchronous Viterbi algorithm for the search process. Both features make it one of the fastest evaluation systems, requiring but a fraction of the computational resources used by its competitors.

November 1994 NAB evaluation Hub1-C1

Two sets of features, one based on Mel-coefficients and one on cepstral coefficients were used. For each feature set 62 feature vectors were computed per second of input speech. This value was not chosen because it gave the smallest word error rate rather than to reduce the recognition time.

The computation of the observation probabilities was performed with context independent recurrent neural networks. To build a speaker independent system, model merging techniques were applied to speaker dependent models.

With this setup the ABBOT system got a 12.4% word error rate for the NAB-C1 (contrast) evaluation conditions.

November 1996 broadcast news evaluation

Again, two sets of features (PLP and Mel-scale) where used, generating 62 feature vectors per second of input speech.

Connectionist models with time delays were trained to estimate posteriori phone probabilities, which were converted to likelihoods and then used instead of the usual observation probabilities in an hidden Markov model decoder. A major change to the 1994 system was in the use of 604 word-internal context dependent phoneme models.

To find the best recognition hypotheses, a single pass decoding with the NOWAY stack decoder was used. With a 65,000 word vocabulary, the error rate of the ABBOT system in the 1996 HUB-4 test was 34.7%. The recognition speed for this

system in the evaluation was estimated to about 60 times slower than real time on an 170MHz UltraSparc, which was considerably faster than most of its competitors.

Selection of Papers

For more information about the ABBOT system, refer to:
[Renals and Hochberg, 1995], [Hochberg et al., 1995b], [Hochberg et al., 1995a], [Cook et al., 1997b], [Cook et al., 1997a], and [Cook et al., 1997c].

A.4 Cambridge University – HTK

With a word error rate of only 10.5% under the NAB-C1 conditions, the HTK system was the best recognizer in the 1994 LVCSR evaluation. Its success appears to be founded on consequently applying all promising algorithms such as quinphone models, cross-word models, 4-gram language models and unsupervised speaker adaptation. Considerable effort was also invested in the pronunciation dictionaries.

Entropics is the commercial spin-off of the Cambridge University HTK system. Since the code commercial code is more stable and more toolkit-oriented, the performance on evaluation sets is a bit lower than of its state-of-the-art cousin. A modified version of the commercial HTK system was also used by the University of Hamburg in the 1996 and 1997 Verbmobil Evaluations [Hübner et al., 1996].

Since the HTK research system is tuned towards maximum performance and the Entropics commercial system towards toolkit flexibility rather than recognition speed, most issues addressed within this thesis have not been pursued for these systems.

November 1994 NAB evaluation Hub1-C1

12 Mel-Cepstral coefficients and their first and second derivatives were used for preprocessing in this evaluation. Cepstral-Mean normalization was performed for each sentence. The acoustic models used were cross-word-allophone HMM models with a maximum context of two phonemes to each side.

Each time synchronous search pass created a word-lattice, that could be re-scored or used to restrict the next search pass. The bigram language model for the first pass was also coded into such a lattice, eliminating the need for an explicit implementation of the language model. Between the first and the second pass, an unsupervised speaker adaptation based on the hypothesis of the first pass was used to reduce the difference between the general speaker independent system to the current speaker.

November 1996 broadcast news evaluation

For preprocessing, the signal power, 12 modified PLP cepstral parameters and their first and second derivatives were used. The acoustic model consisted of gender dependent continuous mixture density hidden Markov models with decision tree clustered triphones and quinphones. Several model sets were used for different speaking styles and recording conditions.

The search algorithm was guided by a lattice which encodes the language model, and generates a lattice that can be used to restrict further search passes. Between search passes, unsupervised speaker adaptation was applied to improve the acoustic models.

Triphone acoustic models and bigram language models were used in first two decoding and MLLR passes. For subsequent passes, quinphones and 4-grams were used.

Using a 65,000 word vocabulary, a 27.5% word error rate was achieved in the November 1996 Broadcast News HUB-4 evaluation. The recognition speed for all passes was estimated to be about 340 times slower than real-time.

Selection of Papers

For more information about the HTK system, refer to: [Woodland et al., 1995], [Knill et al., 1996], [Woodland et al., 1997a], [Humphries and Woodland, 1997], and [Woodland et al., 1997b].

A.5 Carnegie Mellon University: SPHINX

The SPHINX-II system was very successful in a number of (D)ARPA LVCSR evaluations. Its success is due to the early use of context dependent acoustic models tied at the state level. The most frequently used search strategy is a time-synchronous forward pass to find likely word-end points, followed by a time-synchronous backward pass to find likely word-begin points. The result of these two passes is then combined using an A* search. Speed issues for the computation of observation probabilities are addressed by using a very small number of codebooks. The SPHINX system also makes extensive use of integer arithmetic, though on modern systems with multiple floating point units that effort is usually no longer needed. In recent years most of the research in the SPHINX group was aiming at robust speech recognition.

November 1994 NAB evaluation Hub1-C1

Sphinx-II used senonic semi-continuous hidden Markov Models to model context-dependent phones, including those between words. The system used four types of

codebooks: Mel-frequency cepstral coefficients, first cepstral derivative, second cepstral derivative, and the signals power along with its first and second derivative. A number of phone classes were identified (27 for males, 28 for females), and a set of four codebooks was trained for each phone class. Cepstral vectors were normalized with an utterance-based cepstral mean value. The semi-continuous observation probability was computed using a variable-sized mixture of the top four Gaussians from each phone-dependent codebook. This combination of semi-continuous hidden Markov Models with phone-dependent codebooks approached the functionality of continuous hidden Markov models in a computationally feasible fashion. Four sets of models were trained – two female and two male.

In the 1994 HUB1-C1 (NAB, contrast conditions) evaluation, the SPHINX-II-system reached an error rate of 13.7%.

Selection of Papers

For more information about the CMU SPHINX system, refer to:
[Lee and Alleva, 1991], [Alleva et al., 1992], [Huang et al., 1992],
[Alleva et al., 1993], [Hwang et al., 1993b], [Hwang et al., 1993a],
[Chase et al., 1995], [Ravishankar, 1996], and [Ravishankar et al., 1997].

A.6 Daimler Benz

Though not represented in any of the official (D)ARPA LVCSR evaluations, the speech group of Daimler Benz has developed a robust speech recognition component that has been used for the Verbmobil Evaluations. Focus of the research has been on robustness in adverse environments, and little has been published about the internal structure of the decoder.

Apparently, it uses a decoder that organizes the dictionary into a tree and takes advantage of delayed bigrams for efficient language modelling. It can also efficiently produce compact word-graphs as an alternative recognition output.

Selection of Papers

For more information about the Daimler recognizer, refer to: [Class et al., 1993] and [Ehrlich et al., 1997].

A.7 Dragon

Probably the most important feature of the Dragon LVCSR research is, that it has recently led to a commercially viable speaker dependent LVCSR product, *Dragon*

Naturally Speaking. In the 1994 HUB1-C1 (NAB, contrast conditions) evaluation, the Dragon research system reached an error rate of 13.2%. Since a lot of the research has been done with a product in mind, the systems used at Dragon usually require a lot less resources than other research systems competing in the ARPA evaluations. For instance, while most systems were processing 100 feature vectors per second of input speech, Dragon decided to use a coarser sampling of only 50 feature vectors for testing in the 1994 evaluation. While most systems use 4-byte floats for their input features, Dragon used 1-byte parameters. To reduce the resource requirements even more, the commercial version applies automatic vocabulary switching to restrict the search space.

Selection of Papers

For more information about the Dragon research system, refer to:
[Ellermann et al., 1993], [Chevalier et al., 1995] and [Peskin et al., 1997].

A.8 IBM

The IBM speech recognition system differs form other competing systems by the early use of a rank-based approach for the computation of observation probabilities that allows to avoid certain search problems related to extreme probability values. The search strategy is a combination of an A* with a time-synchronous Viterbi search and therefore difficult to compare to the fully time-synchronous search of other systems.

A commercial LVCSR dictation system, *IBM ViaVoice*, has been derived from the research system used for the (D)ARPA evaluations. This system requires several minutes of read speech to adapt to a new user.

November 1994 NAB evaluation Hub1-C1

The preprocessing for this system was one of the few to use a linear discriminant analysis at this time. The original input features were *super-vectors* composed of up to 9 vectors of 24 Mel cepstral coefficients. Using the LDA, these were mapped into a 60 dimensional feature space for the recognizer.

Other features that made this system substantially different from its competitors include that during the first recognition pass the usual mixture of Gaussian probabilities were replaced by a probability derived from the rank of the models score. Also, IBM developed it's own search strategy, the so called *envelope search*, that combines aspects of the A* heuristic search algorithm with those of the time-synchronous Viterbi search.

The error rate of the IBM system in the 1994 HUB-1 NAB-C1 test was 11.1%, putting the system on the second place after the HTK system.

November 1996 broadcast news evaluation

For the 1996 broadcast news evaluation, acoustic models for 5741 sub-phonetic units with context-dependent tying were used. Five separate such acoustic models were trained for different speaking styles and recording conditions.

With 65,000 words in the recognition vocabulary, the IBM-system achieved a 32.2% error rate in official 1996 Broadcast news HUB-4 evaluations.

Selection of Papers

For more information about the IBM research system, refer to:
[Gopalakrishnan et al., 1994], [Bahl et al., 1995a], [Bahl et al., 1995b], [Gopalakrishnan et al., 1995], [Bakis et al., 1997a], [Padmanabhan et al., 1997], and [Bakis et al., 1997b].

A.9 LIMSI

The speech group of the French *Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur* took part in a number of the (D)ARPA evaluations. Their system had the lowest error rates for the November 1993 WSJ and the November 1996 broadcast news evaluation. The system is tuned towards minimum error rate recognition and includes few algorithms to improve the recognition speed.

November 1994 NAB evaluation Hub1-C1

The recognizer used continuous density HMMs with Gaussian mixtures for acoustic modeling and n-gram statistics estimated on newspaper texts for language modeling. A first search pass generated a word-lattice by means of a simple time-synchronous graph-search strategy, which was shown to still be viable with large vocabularies when used with bigram back-off language models. A second forward pass incorporated the trigram language model probabilities.

The gender dependent acoustic modeling was based on cepstrum-based features and consisted of context-dependent phone models (intra and inter-word) with phone duration models. Also, LIMSI did a lot of work to increase the accuracy and consistency of the pronunciation dictionaries for training and testing their system.

In the 1994 HUB1-C1 (NAB, contrast conditions) evaluation, the LIMSI-system reached an error rate of 12.1%.

November 1996 broadcast news evaluation

The LIMSI November 1996 speech recognizer for the ARPA broadcast news task made use of continuous density hidden Markov models with Gaussian mixture for acoustic modeling and trigram statistics estimated on newspaper texts (160M words) and broadcast news transcription (132M words).

The gender dependent acoustic models were trained on the WSJ0/WSJ1, and adapted using MAP estimation with 35 hours of broadcast news data. Also, two sets of models were used to allow for the different speaking styles. Throughout the system, position dependent triphones were used.

The recognizer uses a tree pass search, with more accurate acoustic and language model in successive passes. The first pass created a word graph, using only bigram language models in the process. The second pass re-scored this word graph with a trigram language model. The resulting 1-best hypothesis was used for MLLR adaptation. The resulting acoustic models were used for a final recognition, again with trigram language models.

With a vocabulary of 65,000 words, a 27.1% error rate was obtained in the official 1996 Broadcast news HUB-4 evaluation. The estimated recognition speed during the evaluation was reported to be about 250 to 300 times slower than real-time.

Selection of Papers

For more information about the LIMSI system, refer to: [Gauvain et al., 1993], [Lamel and Gauvain, 1993], [Gauvain et al., 1995b], [Gauvain et al., 1995a], and [Gauvain et al., 1997].

A.10 Microsoft – Whisper

Though not represented in the official (D)ARPA evaluations, there is a significant amount of LVCSR related research and development at Microsoft. Many of their researchers are former CMU students. The few available publications suggest, that their research on LVCSR-systems for fast Pentium PC's allows near real time recognition with very low error rates for dictation applications.

Selection of Papers

For more information about the Microsoft Whisper LVCSR system, refer to [Huang et al., 1995] and [Alleva et al., 1997].

A.11 MIT – Lincoln Laboratories

The most important differences between the MIT-System and most other LVCSR systems are, that the MIT system is using a stack decoder instead of a time-synchronous search, and that the number of people working on it is substantially smaller. Little new work was reported on the development of LVCSR systems since the 1994 NAB evaluation.

The word error rate of the MIT-system in the 1994 HUB-1 NAB-C1 evaluation was 19%.

Selection of Papers

For more information about the Lincoln Stack-Decoder, refer to [Paul, 1992b], [Paul and Neioglu, 1993],[Paul, 1995a],[Paul, 1995b], and [Paul, 1995c].

A.12 Philips and University of Aachen

The group around Hermann Ney, previously at Philips Research Labs and now at the University of Aachen, has a long history of research of fast speech recognition techniques. Most of the research is focused on language modelling and the search algorithm. They have pursued a variety of approaches to build word graphs and find the best recognition hypothesis, some of which resemble the algorithms used in JRTk. However, recent papers also explore possibilities for fast observation probability computation. The latest published recognition times for the 1994 NAB set with a vocabulary size of 20,000 words are around 8 to 16 times real time for error rates over 16%.

November 1994 NAB evaluation Hub1-C1

The system built for the November 1994 NAB evaluation used a multi-pass recognition approach. First, high density word graphs were produced using a time synchronous search with a fast approximation bigram language model. Further passes were guided by this word graph to include full bigram information. The resulting word graph was re-scored using a trigram language model and unsupervised speaker adaptation of the acoustic models. For acoustic modelling, clustered within-word triphones were applied.

In the 1994 HUB1-C1 (NAB, contrast conditions) evaluation, the Philips-system reached an error rate of 13.4%.

Selection of Papers

For more information about the speech recognition systems of these two groups, refer to [Ney, 1984], [Ney and Essen, 1991], [Ney et al., 1992], [Oerder and Ney, 1993], [Ney, 1993], [Steinbiss et al., 1993], [Ney and Essen, 1993], [Steinbiss et al., 1994], [Bugast et al., 1995], [Ortmanns et al., 1997a], [Ortmanns et al., 1997b], and [Ney et al., 1997].

A.13 Siemens

Though not represented in the official ARPA or Verbmobil evaluations, the speech group of the Siemens research laboratories have developed an LVCSR system that achieves real time performance on Pentium PCs by using special purpose hardware for the computation of observation probabilities. Therefore, the complexity of the acoustic models used for the computation of these observation probabilities can be very large, yielding a high discriminative power that helps to reduce the remaining computational load small by pruning the search space.

Selection of Papers

For more information about this research system, refer to [Hauenstein, 1993] and [Niemöller et al., 1997].

A.14 SRI – DECIPHER

The SRI system is another example for a recognizer using a multi-pass time synchronous search, with tied mixture hidden Markov models. Most of the development effort went into the reduction of the error rate, and only little research was reported on means for achieving real time recognition.

November 1994 NAB evaluation Hub1-C1

The front-end processing extracted 6 spectral features to model the speech signal: cepstral coefficients and the normalized energy, both with first and second derivatives. The cepstral features were computed from an FFT filter-bank and subsequent cepstral-mean normalization on a sentence basis is performed. The 6 spectral features were modeled as a single 39-dimensional observation stream.

The multi-pass-search scheme used for this evaluation has the following passes: a first forward-backward pass generates word-lattices using a 60,000-word bigram language model and context-dependent, genonic hidden Markov models. Only within-word context dependent models were used in the first pass. The first-pass genonic HMMs

have 1,800 Gaussian mixtures per gender, each with 32 components. These Gaussian codebooks (genones) were shared among disjoint sets of HMM states that were determined automatically using clustering techniques. The Gaussian computation was sped up using vector quantization and Gaussian shortlists. Both forward and backward passes used recognition networks with a tree-structured back-off node.

The second pass performed a forward-backward N-best search on the word-lattices using the first-pass genonic hidden Markov models. The N-best lists were then re-scored using more expensive acoustic and language models.

In the 1994 HUB1-C1 (NAB, contrast conditions) evaluation, the SRI-system reached an error rate of 12.2%.

Selection of Papers

For further information about the SRI system, refer to [Murveit and Butzberger, 1992],[Murveit et al., 1993],[Digalakis et al., 1995] and [Weng et al., 1997].

A.15 Other LVCSR Systems

Several smaller groups, such as the speech groups at Boston University [Ostendorf et al., 1995] and New York University [Sekine et al., 1995] cooperated with the larger groups at BBN and SRI for the ARPA LVCSR evaluations, often by producing results in the form of re-scored N-best lists.

There are also research groups focusing on languages other than English. For instance, NTT does intensive research in Japanese LVCSR and has published a number of papers on this topic [Matsuoka et al., 1997].

This list is by no means complete, but gives only a brief glimpse at the LVCSR research community.

Bibliography

[Alleva et al., 1992] Alleva, F., Hon, H., Huang, X., Hwang, M., and Rosenfeld, R. Applying Sphinx–II to the DARPA Wall Street Journal CSR Task. In *Proceedings of the DARPA 1992 [DARPA Workshop, 1992].*

[Alleva et al., 1993] Alleva, F., Huang, X., and Hwang, M.-Y. An Improved Search Algorithm Using Incremental Knowledge for Continuous Speech Recognition. In *[ICASSP, 1993]*, volume 2, pages 307–310.

[Alleva et al., 1997] Alleva, F., Huang, X., and Hwang, M.-Y. Improvements on the Pronunciation Prefix Tree Search Organisation. In *[ICASSP, 1996]*, volume 1, pages 133–136.

[Appelt and Jackson, 1992] Appelt, D. E. and Jackson, E. SRI International February 1992 ATIS Benchmark Test Results. In *Proceedings of the DARPA 1992 [DARPA Workshop, 1992].*

[ARPA Workshop, 1995] *Proceedings of the ARPA Spoken Language Systems Technology Workshop*, January, 1995, Austin, Texas.

[Aubert, 1995] Aubert, X. Large Vocabulary Continuous Speech Recognition Using Word Graphs. In *[EUROSPEECH, 1995]*, volume 1, pages 49–52.

[Austin et al., 1992] Austin, S., Schwartz, R., and Makhoul, J. Speech Recognition Using Segmental Neural Nets. In *[ICASSP, 1992]*, volume 1, pages 625–628.

[Austin et al., 1991] Austin, S., Schwartz, R., and Placeway, P. The Forward–Backward Search Algorithm. In *[ICASSP, 1991]*, volume 1, pages 697–700.

[Bahl et al., 1995a] Bahl, L., Balakrishnan-Aiyer, S., Bellegarda, J., Franz, M., Gopalakrishnan, P., Nahamoo, D., Novak, M., Padmanabhan, M., Picheny, M., and Roukos, S. Performance of the IBM Large Vocabulary Continuous Speech Recognition System on the ARPA Wall Street Journal Task. In *[EUROSPEECH, 1995]*, volume 1, pages 41–44.

[Bahl et al., 1993] Bahl, L., de Souza, P., Gopalakrishnan, P., Nahamoo, D., and Picheny, M. Word Lookahead Scheme for Cross-Word Right Context Models in a Stack Decoder. In *[EUROSPEECH, 1993]*, volume 2, pages 851–854.

[Bahl et al., 1995b] Bahl, L. R., Balakrishnan-Aiyer, S., Franz, M., Gopalakrshnan, P., Gopinath, R., Novak, M., Padmanabhan, M., and Roukos, S. Performance of the IBM Large Vocabulary Continuous Speech Recognition System on the ARPA NAB News Task. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Bakis et al., 1997a] Bakis, R., Chen, S., Gopalakrishnan, P., Gopinath, R., Mes, S., and Polymenakos, L. Transcription of broadcast news shows with the IBM Large Vocabulary Speech Recognition System. In *DARPA 1997 Speech Recognition Workshop.*

[Bakis et al., 1997b] Bakis, R., Schen, S., Gopalakrishnan, P., Gopinath, R., Maes, S., and Polymenakos, L. Transcription of Broadcast News - System Robustness Issues and Adaptation Techniques. In *[ICASSP, 1997]*, volume 2, pages 711–715.

[Bamberg and Baur, 1996] Bamberg, G. and Baur, F. *Statistik.* Oldenbourg, R.Oldenbourg Verlag München Wien. ISBN 3-486-23520-6.

[Bei and Gray, 1985] Bei, C. and Gray, R. An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization. In *IEEE Transactions on Communications, Vol. Com-33*, pages 1132–1133.

[Bentley, 1975] Bentley, J. Multidimensional binary search trees used for associative searching. In *Commun. Ass. Comput. Mach*, volume 18, no 9, pages 509–517.

[Bocchieri, 1993a] Bocchieri, E. A Study of the Beam-Search Algorithm for Large Vocabulary Continuous Speech Recognition and Methods for Improved Efficiency. In *[EUROSPEECH, 1993]*, volume 3, pages 1521–1524.

[Bocchieri, 1993b] Bocchieri, E. Vector Quantisation for the Efficient Computation of Continuous Density Likelyhoods. In *[ICASSP, 1993]*, volume 2, pages 692–695.

[Bocchieri and Mak, 1997] Bocchieri, E. and Mak, B. Subspace Distribution Clustering for Continuous Observation Density Hidden Markov Models. In *[EUROSPEECH, 1997]*, volume 1, pages 107–110.

[Boiteau and Haffner, 1993] Boiteau, D. and Haffner, P. Connectionist Segmental Post-Processing of the N-Best Solutions in Isolated and Connected Word Recognition Task. In *[EUROSPEECH, 1993]*, volume 3, pages 1933–1937.

[Bub et al., 1997] Bub, T., Wahlster, W., and Waibel, A. VERBMOBIL: The Combination Of Deep and Shallow Processing For Spontaneous Speech Translation. In *[ICASSP, 1997]*, volume 1, pages 71–74.

[Bugast et al., 1995] Bugast, C., Kneser, R., Aubert, X., Orthmanns, S., Beulen, K., and Ney, H. Continuous Speech Recognition Tests and Results for the NAB'94 Corpus. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Chase et al., 1995] Chase, L., Rosenfeld, R., Hauptmann, A., Ravishankar, M., Thayer, E., Placeway, P., R.Weide, and Lu, C. Improvements in Language, Lexical and Phonetic Modeling in SPHINX-II. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Chevalier et al., 1995] Chevalier, H., Ingold, C., Kunz, C., Moore, C., Roven, C., Yamron, J., Baker, B., Bamberg, P., Bridle, S., Bruce, T., and Weader, A. Large-vocabulary Speech Recognition in Specialized Domains. In *[ICASSP, 1995]*, volume 1, pages 217–220.

[Class et al., 1993] Class, F., Kaltenmeier, A., and Regel-Brietzmann, P. Optimization of an HMM-Based Continuous Speech Recognizer. In *[EUROSPEECH, 1993]*, volume 2, pages 803–806.

[Cook et al., 1997a] Cook, C. D., Kershaw, D. J., Christie, J. D. M., Seymour, C. W., and Waterhouse, S. R. Transcription of Broadcast Television and Radio News: The 1996 Abbot System. In *[ICASSP, 1997]*, volume 2, pages 723–727.

[Cook et al., 1997b] Cook, G., Kershaw, D., Christie, J. D. M., and Robinson, A. J. Transcription of Broadcast television and radio news: the 1996 ABBOT system. In *DARPA 1997 Speech Recognition Workshop.*

[Cook et al., 1997c] Cook, G. D., Waterhouse, S. R., and Robinson, A. J. Ensemble Methods for Connectionist Acoustic Modelling. In *[EUROSPEECH, 1997]*, volume 4, pages 1959–1962.

[DARPA Workshop, 1992] *Fifth DARPA Workshop on Speech and Natural Language*, February, 1992, Washington, DC.

[Digalakis et al., 1995] Digalakis, V., Weintraub, M., Sankar, A., Franco, H., Neumeyer, L., and Murveit, H. Continuous speech dictation on ARPAs NAB News Domain. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Ehrlich et al., 1997] Ehrlich, U., Hanrieder, G., Hitzenberger, L., Heisterkamp, P., Mecklenburg, K., and Regel-Brietzmann, P. ACCeSS - Automated Call Center Through Speech Understanding System. In *[EUROSPEECH, 1997]*, volume 4, pages 1819–1822.

[Ellermann et al., 1993] Ellermann, C., Even, S. V., Huang, C., and Manganaro, L. Dragon Systems' Experiences in Small to Large Vocabulary Multi-Lingual Speech Recognition Applications. In *[EUROSPEECH, 1993]*, volume 3, pages 2077–2080.

[EUROSPEECH, 1989] *Proceedings of the 1st European Conference on Speech, Communication and Technology*, September, 1989, Paris, France.

[EUROSPEECH, 1991] *Proceedings of the 2nd European Conference on Speech, Communication and Technology*, 1991, Genova, Italy.

[EUROSPEECH, 1993] *Proceedings of the 3rd European Conference on Speech, Communication and Technology*, September, 1993, Berlin, Germany.

[EUROSPEECH, 1995] *Proceedings of the 4th European Conference on Speech, Communication and Technology*, September, 1995, Madrid, Spain.

[EUROSPEECH, 1997] *Proceedings of the 5th European Conference on Speech, Communication and Technology*, September, 1997, Rhodes, Greece.

[Finke and Rogina, 1997] Finke, M. and Rogina, I. Wide Context Acoustic Modeling in Read vs. Spontaneous Speeech. In *[ICASSP, 1997]*, volume 3, pages 1743–1746.

[Fisher and Fiscus, 1993] Fisher, W. and Fiscus, J. Better Alignment Procedures for Speech Recognition Evaluation. In *[ICASSP, 1993]*, volume 2, pages 59–62.

[Fissore et al., 1993] Fissore, L., Laface, P., Massafra, P., and Ravera, F. Analysis and Improvement of the Partial Distance Search Algorithm. In *[ICASSP, 1993]*, volume 2, pages 315–318.

[Fritsch et al., 1997] Fritsch, J., Finke, M., and Waibel, A. Context-Dependent Hybrid HME/HMM Speech Recognition Using Polyphone Clustering Decision Trees. In *[ICASSP, 1997]*, volume 3, pages 1759–1762.

[Fritsch and Rogina, 1996] Fritsch, J. and Rogina, I. The Bucket Box Intersection (BBI) Algorithm for fast approximative evaluation of Diagonal Mixture Gaussians. In *[ICASSP, 1996]*, volume 1, pages 837–840.

[Fritsch et al., 1995] Fritsch, J., Rogina, I., Sloboda, T., and Waibel, A. Speeding up the Score Computation of HMM Speech Recognizers with the Bucket Voronoi Intersection Algorithm. In *[EUROSPEECH, 1995]*, volume 2, pages 1091–1094.

[Fukunaga, 1990] Fukunaga, K. *Introduction to Statistical Pattern Recogniton.* Academic Press. ISBN 0-12-269851-7.

[Garudadri et al., 1994] Garudadri, H., Labute, P., Boulianne, G., and Kenny, P. Fast Match Acoustic Models in Large Vocabulary Continuous Speech Recognition. In *[ICASSP, 1994]*, volume 2, pages 157–160.

[Gauvain et al., 1993] Gauvain, J., Adda, L. L. G., and Adda-Decker, M. Speaker-Independent Continuous Speech Dictation. In *[EUROSPEECH, 1993]*, volume 1, pages 125–128.

[Gauvain et al., 1997] Gauvain, J. L., Adda, G., Lamel, L., and Adda-Decker, M. Transcribing Broadcast News Shows. In *[ICASSP, 1997]*, volume 2, pages 715–719.

[Gauvain et al., 1995a] Gauvain, J. L., Lamel, L., and Adda-Decker, M. Developments in Continuous Speech Dictation using the ARPA WSJ Task. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Gauvain et al., 1995b] Gauvain, J. L., Lamel, L., and Adda-Decker, M. Developments in Continuous Speech Dictation Using the ARPA WSJ Task. In *[ICASSP, 1995]*, volume 1, pages 65–68.

[Gilick and Coz, 1989] Gilick, L. and Coz, S. Some Statistical Issues in the Comparision of Speech Recognition Algorithms. In *[ICASSP, 1989]*, volume 1, pages 532–535.

[Gong et al., 1991] Gong, Y., Hanton, J., and Mouria, F. Continuous Speech Recogniton Based on High Plausibility Regions. In *[ICASSP, 1991]*, volume 1, pages 725–728.

[Gopalakrishnan et al., 1995] Gopalakrishnan, P., Bahl, L., and Mercer, R. A tree search strategy for large vocabulary continuous speech recognition. In *[ICASSP, 1995]*, volume 1, pages 572–575.

[Gopalakrishnan et al., 1994] Gopalakrishnan, P., Nahamoo, D., Padmanabhan, M., and Pincheny, M. A. A Channel-Bank-Based Phone Detection Strategy. In *[ICASSP, 1994]*, volume 2, pages 161–164.

[Hauenstein, 1993] Hauenstein, A. Architecture of a 10000 Word Real Time Speech Recognizer. In *[EUROSPEECH, 1993]*, volume 3, pages 1829–1832.

[Hauptmann and Wactlar, 1997] Hauptmann, A. and Wactlar, H. Indexing and Search of Multimodal Information. In *[ICASSP, 1997]*, volume 1, pages 195–198.

[Hetherington et al., 1993] Hetherington, I. L., Phillips, M. S., Glass, J. R., and Zue, V. W. A* Word Network Search for Continuous Speech Recognition. In *[EUROSPEECH, 1993]*, volume 3, pages 1533–1536.

[Hochberg et al., 1995a] Hochberg, M., Renals, S., Robinson, A. J., and Cook, G. D. Recent Improvements to the ABBOT Large Vocabulary CSR System. In *[ICASSP, 1995]*, volume 1, pages 69–72.

[Hochberg et al., 1995b] Hochberg, M. M., Cook, G. D., Renals, S. J., Robinson, A. J., and Schechtman, R. S. The 1994 ABBOT Hybrid connectionist-hmm Large-Vocabulary Recognition System. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Huang et al., 1995] Huang, X., Acero, A., Alleva, F., Hwang, M., Jiang, L., and Mahajan, M. Microsoft Windows Highly Intelligent Speech Recognizer: Whisper. In *[ICASSP, 1995]*, volume 1, pages 93–97.

[Huang et al., 1992] Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y., and Rosenfeld, R. The SPHINX–II Speech Recognition System. Technical Report CS-92-112, Carnegie Mellon University (USA).

[Hübner et al., 1996] Hübner, K., Jost, U., and Heine, H. Speech Recognition for Spontaneously Spoken German Dialogues. In *[ICSLP, 1996]*, volume 1, pages 212–215.

[Humphries and Woodland, 1997] Humphries, J. J. and Woodland, P. C. Using Accent-Specific Pronunciation Modelling For Improved Large Vocabulary Continuous Speech Recognition. In *[EUROSPEECH, 1997]*, volume 5, pages 2367–2370.

[Hwang et al., 1993a] Hwang, M.-Y., Alleva, F., and Huang, X. Senones, Multi-Pass Search, and Unified Stochastic Modeling in SPHINX-II. In *[EUROSPEECH, 1993]*, volume 3, pages 2143–2146.

[Hwang and Huang, 1992] Hwang, M.-Y. and Huang, X. Subphonetic Modeling with Markov States – Senone. In *[ICASSP, 1992]*, volume 1, pages 33–37.

[Hwang et al., 1993b] Hwang, M.-Y., Huang, X., and Alleva, F. Predicting unseen Triphones with Senones. In *[ICASSP, 1993]*, volume 2, pages 311–314.

[ICASSP, 1988] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April, 1988, New York, USA. IEEE.

[ICASSP, 1989] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, May, 1989, Glasgow, GB. IEEE.

[ICASSP, 1990] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April, 1990, Albuquerque, USA. IEEE.

[ICASSP, 1991] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, May, 1991, Toronto, Canada. IEEE.

[ICASSP, 1992] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, March, 1992, San Francisco, USA. IEEE.

[ICASSP, 1993] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April, 1993, Minneapolis, USA. IEEE.

[ICASSP, 1994] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April, 1994, Adelaide, Australia. IEEE.

[ICASSP, 1995] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, May, 1995, Detroit, USA. IEEE.

[ICASSP, 1996] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, May, 1996, Atlanta, USA. IEEE.

[ICASSP, 1997] *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, April, 1997, München, Germany. IEEE.

[ICSLP, 1994] *Proceedings of the Third International Conference on Spoken Language Processing*, September, 1994, Yokohama, Japan.

[ICSLP, 1996] *Proceedings of the Fourth International Conference on Spoken Language Processing*, October, 1996, Philadephia, USA.

[Jelinek, 1990] Jelinek, F. Self-Organized Language Modeling for Speech Recognition. In *Readings in Speech Recognition [Waibel and Lee, 1990]*, pages 450–506. Morgan Kaufmann.

[Jiménez et al., 1993] Jiménez, V. M., Marzal, A., and Vidal, E. Efficient Enumeration of Sentence Hypotheses in Connected Word Recognition. In *[EUROSPEECH, 1993]*, volume 3, pages 2183–2186.

[Jürgens, 1995] Jürgens, M. Implementierung eines Cluster-Algorithmus für Codebücher auf der MASPAR MP-1. Studienarbeit.

[Katz, 1987] Katz, S. M. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume ASSP-35, pages 400–401.

[Kenny et al., 1993] Kenny, P., abd Z. Li, P. L., Hollan, R., Lenning, M., and O'Shaughnessy, D. A New Fast Match For Very Large Vocabulary Continuous Speech Recognition. In *[ICASSP, 1993]*, volume 2, pages 656–659.

[Knill et al., 1996] Knill, K. M., Gales, M. J. F., and Young, S. J. Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition using HMMs. In *[ICSLP, 1996]*, volume 1, pages 470–473.

[Kubala et al., 1997a] Kubala, F., Jin, H., Matsoukas, S., and Nguyen, L. The 1996 BBN Byblos HUB-4 Transcription System. In *DARPA 1997 Speech Recognition Workshop*.

[Kubala et al., 1997b] Kubala, F., Jin, H., Matsoukas, S., Nguyen, L., Schwartz, R., and Makhoul, J. Advances in Transcription of Broadcast News . In *[EUROSPEECH, 1997]*, volume 2, pages 927–930.

[Kubala et al., 1997c] Kubala, F., Jin, H., Nguyen, L., Schwartz, R., and Matsoukas, S. Broadcast News Transcription. In *[ICASSP, 1997]*, volume 1, pages 203–207.

[Kurimo and Somervuo, 1996] Kurimo, M. and Somervuo, P. Using the Self-Organizing Map to Speed up the Probability Density Estimation for Speech Recognition with Mixture Density HMMS. In *[ICSLP, 1996]*, volume 1, pages 358–361.

[Lacouture and Normandin, 1993] Lacouture, R. and Normandin, Y. Efficient Lexical Access Strategies. In *[EUROSPEECH, 1993]*, volume 3, pages 1537–1540.

[Lamel and Gauvain, 1993] Lamel, L. F. and Gauvain, J. High Performance Speaker-Independent Phone Recognition Using CDHMM. In *[EUROSPEECH, 1993]*, volume 1, pages 121–124.

[Lee, 1988] Lee, K. *Large–vocabulary Speaker–independent Continuous Speech Recognition: The Sphinx System.* PhD thesis, Carnegie Mellon University.

[Lee and Hon, 1988] Lee, K. and Hon, W. Large Vocabulary Speaker–Independent Continous Speech Recognition. In *[ICASSP, 1988]*.

[Lee and Alleva, 1991] Lee, K.-F. and Alleva, F. Continuous Speech Recognition. In Sohnhi, M. M. and Furi, S., editors, *Advances in Speech Signal Processing.* Pub. Marcel Dekker.

[Ljolje et al., 1995] Ljolje, A., Riley, M., Hindle, D., and Pereira, F. The AT&T 60,000 Word Speech-To-Text System. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995]*.

[Matsuoka et al., 1997] Matsuoka, T., Ohtsuki, K., Mori, T., Yoshida, K., Furui, S., and Shirai, K. Japanese Large-Vocabulary Continuous-Speech Recognition using a Business-Newspaper Corpus. In *[ICASSP, 1997]*, volume 3, pages 1803–1807.

[McNair and Waibel, 1994] McNair, A. and Waibel, A. Improving Recognizer Acceptance through robust, natural Speech Repair. In *[ICSLP, 1994]*, volume 4, pages 1299–1302.

[Mohri and Riley, 1997] Mohri, M. and Riley, M. Weighted Determinization and Minimization for Large Vocabulary Speech Recognition. In *[EUROSPEECH, 1997]*, volume 1, pages 131–134.

[Murveit and Butzberger, 1992] Murveit, H. and Butzberger, J. Performance of SRI's Decipher Speech Recoginition System on DARPA's CSR Task. In *Proceedings of the DARPA 1992 [DARPA Workshop, 1992]*.

[Murveit et al., 1993] Murveit, H., Butzberger, J., Digalakis, V., and Weintraub, M. Large-Vocabulary Dictation using SRI's DECIPHER Speech Recognition System: Progressive Search Techniques. In *[ICASSP, 1993]*, volume 2, pages 319–322.

[Ney, 1984] Ney, H. The Use of a One–Stage Dynamic Programming Algorithm for Connected Word Recognition. In *Readings in Speech Recognition [Waibel and Lee, 1990]*, pages 188–196. Morgan Kaufmann.

[Ney, 1993] Ney, H. Modeling and Search in Continuous Speech Recognition. In *[EUROSPEECH, 1993]*, volume 1, pages 491–498.

[Ney and Essen, 1991] Ney, H. and Essen, U. On Smoothing Technique For Bigram–Based Language Modeling. In *[ICASSP, 1991]*, volume 2, pages 825–828.

[Ney and Essen, 1993] Ney, H. and Essen, U. Estimating Small Probabilities by Leaving-One-Out. In *[EUROSPEECH, 1993]*, volume 3, pages 2239–2242.

[Ney et al., 1992] Ney, H., Haeb-Umbach, R., Tran, B., and M.Oerder. Improvements in Beam Search for 10.000–Word Continuous Speech Recognition. In *[ICASSP, 1992]*, volume 1, pages 9–12.

[Ney et al., 1997] Ney, H., Ortmanns, S., and Lindam, I. Extensions to the Word Graph Method for Large Vocabulary Continuous Speech Recognition. In *[ICASSP, 1997]*, volume 3, pages 1791–1794.

[Nguyen et al., 1995] Nguyen, L., Makhoul, J., Schwartz, R., Kubala, F., LaPre, C., Yuan, N., Zhao, Y., Anastakos, T., and Zavaliagkos, G. The 1994 BBN/BYBLOS Speech Recognition System. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Nguyen and Schwartz, 1997a] Nguyen, L. and Schwartz, R. Efficient 2-Pass N-Best Decoder. In *[EUROSPEECH, 1997]*, volume 1, pages 167–170.

[Nguyen and Schwartz, 1997b] Nguyen, L. and Schwartz, R. Efficient 2-Pass N-Best Decoder . In *DARPA 1997 Speech Recognition Workshop.*

[Nguyen et al., 1994] Nguyen, L., Schwartz, R., Zhao, Y., and Zavaliagkos, G. Is N-Best Dead. In *Proceedings of the DARPA 1994 Human Language Technology Workshop.*

[Niemöller et al., 1997] Niemöller, M., Hauenstein, A., Marschall, E., Witschel, P., and Harke, U. A PC-Based Real-Time Large Vocabulary Continuous Speech Recognizer for German. In *[ICASSP, 1997]*, volume 3, pages 1807–1810.

[Oerder and Ney, 1993] Oerder, M. and Ney, H. Word Graphs: An Efficient Interface between Continuous-Speech Recognition and Language Understanding. In *[ICASSP, 1993]*, volume 2, pages 119–122.

[Ortmanns et al., 1997a] Ortmanns, S., Eiden, A., Ney, H., and Coenen, N. Look-Ahead Techniques for Fast Beam Search. In *[ICASSP, 1997]*, volume 3, pages 1783–1786.

[Ortmanns et al., 1997b] Ortmanns, S., Firzlaff, T., and Ney, H. Fast Likelihood Computation Methods for Continuous Mixture Densities in Large Vocabulary Speech Recognition. In *[EUROSPEECH, 1997]*, volume 1, pages 139–142.

[Ostendorf et al., 1995] Ostendorf, M., Richardson, F., Kannan, A., Iyer, R., Ronen, O., and Bates, R. The 1994 BU WSJ Benchmark System. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Osterholtz et al., 1992] Osterholtz, L., McNair, A., Rogina, I., Saito, H., Sloboda, T., Tebelskis, J., Waibel, A., and Woszczyna, M. Testing Generality in JANUS: A Multi–Lingual Speech to Speech Translation System. In *[ICASSP, 1992]*, volume 1, pages 209–212.

[Padmanabhan et al., 1997] Padmanabhan, M., Bahl, L. R., Nahamoo, D., and de Souza, P. Decision-Tree Based Quantization of the Feature Space of a Speech Recognizer. In *[EUROSPEECH, 1997]*, volume 1, pages 147–150.

[Paschen, 1997] Paschen, K. Lookahead mit Neuronalen Netzen in JANUS. Studienarbeit.

[Paul, 1995a] Paul, D. New Developments in the Lincoln Stack-Decoder Based Large Vocabulary CSR System. In *[EUROSPEECH, 1995]*, volume 1, pages 45–48.

[Paul, 1992a] Paul, D. B. An Efficient A∗ Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model. In *[ICASSP, 1992]*, volume 1, pages 25–28.

[Paul, 1992b] Paul, D. B. The Lincoln Large-Vocabulary HMM CSR. In *Proceedings of the DARPA 1992 [DARPA Workshop, 1992].*

[Paul, 1995b] Paul, D. B. New Developments in the Lincoln Stack-Decoder Based Large Vocabulary CSR System. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Paul, 1995c] Paul, D. B. New Developments in the Lincoln Stack-Decoder Based Large Vocabulary CSR System. In *[ICASSP, 1995]*, volume 1, pages 45–49.

[Paul and Neioglu, 1993] Paul, D. B. and Neioglu, B. F. The Lincoln Large-Vocabulary Stack-Decoder HMM CSR. In *[ICASSP, 1993]*, volume 2, pages 660–663.

[Peskin et al., 1997] Peskin, B., Gillick, L., Liberman, N., Newman, M., van Mulbregt, P., and Wegmann, S. Progress in Recognizing Conversational Telephone Speech. In *[ICASSP, 1997]*, volume 3, pages 1811–1814.

[Phillis and Rogers, 1997] Phillis, S. and Rogers, A. Parallel Speech Recognition. In *[EUROSPEECH, 1997]*, volume 1, pages 242–245.

[Rabiner, 1989] Rabiner, L. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Readings in Speech Recognition [Waibel and Lee, 1990]*, pages 267–296. Morgan Kaufmann.

[Ramasubramanian and Paliwal, 1992a] Ramasubramanian, V. and Paliwal, K. Fast K–Dimensional Tree Algorithms for Neares Neighbor Search with Application to Vector Quantization Encoding. *IEEE Transactions on Signal Processing*, pages 518–531.

[Ramasubramanian and Paliwal, 1992b] Ramasubramanian, V. and Paliwal, K. An Efficient Approximation-Elimination Algorithm for Fast Nearest-Neighbor Search. In *[ICASSP, 1992]*, volume 1, pages 89–92.

[Ravishankar et al., 1997] Ravishankar, M., Bisiani, R., and Thayer, E. Sub-Vector Clustering to Improve Memory and Speed Performance of Acoustic Likelihood Computation. In *[EUROSPEECH, 1997]*, volume 1, pages 151–154.

[Ravishankar, 1996] Ravishankar, M. K. *Efficient Algorithms for Speech Recognition.* PhD thesis, Carnegie Mellon University.

[Renals and Hochberg, 1995] Renals, S. and Hochberg, M. Efficient Search Using Posterior Phone Probability Estimates. In *[ICASSP, 1995]*, volume 1, pages 596–599.

[Rogina, 1997] Rogina, I. *Parameterraumoptimierung für Diktiersysteme mit unbeschränktem Vokabular.* PhD thesis, Fakultät für Informatik, Universität Karlsruhe.

[RSI, 1997] RSI. (gbp)186 000 record damages for rsi sufferer at newcastle high court. In *Health and Safety at Work*, volume 19-3.

[Sakoe, 1979] Sakoe, H. Two–Level DP–Matching — A Dynamic Programming–Based Pattern Matching Algorithm for Connected Word Recognition. In *Readings in Speech Recognition [Waibel and Lee, 1990]*, pages 180–187. Morgan Kaufmann.

[Sakoe and Chiba, 1978] Sakoe, H. and Chiba, S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. In *Readings in Speech Recognition [Waibel and Lee, 1990]*, pages 159–165. Morgan Kaufmann.

[Schukat-Talamazzini et al., 1993] Schukat-Talamazzini, E., Bielecki, M., Niemann, H., Kuhn, T., and Rieck, S. A Non-Metrical Space Search Algorithm for Fast Gaussian Vector Quantisation. In *[ICASSP, 1993]*, volume 2, pages 688–691.

[Schwartz and Austin, 1991] Schwartz, R. and Austin, S. A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses. In *[ICASSP, 1991]*, volume 1, pages 701–704.

[Schwartz et al., 1992] Schwartz, R., Austin, S., Kubala, F., Makhoul, J., Nguyen, L., and Placeway, P. New Uses for the N-best Sentence Hypotheses within the BYBLOS Speech Recognition System. In *[ICASSP, 1992]*, volume 1, pages 1–4.

[Schwartz and Chow, 1990] Schwartz, R. and Chow, Y.-L. The N–best Algorithm: An Efficient and Exact Procedure for Finding the N Most Likely Sentence Hypotheses. In *[ICASSP, 1990]*, volume 1, pages 81–84.

[Sekine et al., 1995] Sekine, S., Sterling, J., and Grishman, R. NYU/BBN 1994 CSR Evaluation. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995].*

[Sloboda, 1992] Sloboda, T. Algorithmen der Spracherkennung auf massiv parallelen SIMD-Rechnern. Master's thesis, Fakultät für Informatik, Universität Karlsruhe.

[Steinbiss, 1989] Steinbiss, V. Sentence–Hypotheses Generation in a Continous–Speech Recognition System. In *[EUROSPEECH, 1989]*, volume 2, pages 51–54.

[Steinbiss et al., 1993] Steinbiss, V., Ney, H., Haeb-Umbach, R., Tran, B., Essen, U., Kneser, R., Oerder, M., Meier, H., Aubert, X., Durgast, C., and Geller, D. The Phillips Research System for Large-Vocabulary Continuous Speech Recognition. In *[EUROSPEECH, 1993]*, volume 3, pages 2125–2128.

[Steinbiss et al., 1994] Steinbiss, V., Tran, B.-H., and Ney, H. Improvements in beam search. In *[ICSLP, 1994]*, volume 4, pages 2143–2146.

[Suhm et al., 1993] Suhm, B., Woszczyna, M., and Waibel, A. Detection and Transcription of New Words. In *[EUROSPEECH, 1993]*, volume 3, pages 2179–2182.

[Tebelskis, 1995] Tebelskis, J. *Speech Recognition Using Neural Networks.* PhD thesis, Carnegie Mellon University.

[Wahle, 1994] Wahle, S. Untersuchungen zur Parallelisierbarkeit von Backpropagation-Netzwerken auf dem CNAPS Neurocomputer. Master's thesis, Fakultät für Informatik, Universität Karlsruhe.

[Waibel et al., 1991] Waibel, A., Jain, A., McNair, A., Saito, H., Hauptmann, A., and Tebelskis, J. JANUS: A Speech-to-Speech Translation System Using Connectionist and Symbolic Processing Strategies. In *[ICASSP, 1991]*, volume 2, pages 793–796.

[Waibel and Lee, 1990] Waibel, A. and Lee, K.-F., editors. *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, CA. ISBN 1-558-60124-4.

[Waibel et al., 1997] Waibel, A., Suhm, B., Vo, M. T., and Yang, J. Multimodal Interfaces for Multimedia Information Agents. In *[ICASSP, 1997]*, volume 1, pages 167–170.

[Welch, 1995] Welch, B. B. *Practical Programming in Tcl and Tk.* Prentice Hall PTR, Upper Saddle River, New Jersey 07458. ISBN 0-13-182007-9.

[Weng et al., 1997] Weng, F., Bratt, H., Neumeyer, L., and Stolcke, A. A Study of Multilingual Speech Recognition. In *[EUROSPEECH, 1997]*, volume 1, pages 359–362.

[Woodland et al., 1997a] Woodland, P. C., Gales, M. J. F., Pye, D., and Young, S. Broadcast news Transcription using HTK. In *DARPA 1997 Speech Recognition Workshop.*

[Woodland et al., 1997b] Woodland, P. C., Gales, M. J. F., Pye, D., and Young, S. J. Broadcast News Transcription Using HTK. In *[ICASSP, 1997]*, volume 2, pages 719–723.

[Woodland et al., 1995] Woodland, P. C., Leggetter, C. H., Odell, J. J., Valtchev, V., and Young, S. J. The Development of the 1994 HTK Large Vocabulary Speech Recognition System. In *Proceedings of the ARPA SLT Workshop 1995 [ARPA Workshop, 1995]*.

[World Factbook, 1996] World Factbook. *The World Factbook 1996*. Central Intelligence Agency, Washington, DC.

[Woszczyna et al., 1993] Woszczyna, M., Coccaro, N., Eisele, A., Lavie, A., McNair, A., Polzin, T., Rogina, I., Rose, C., Sloboda, T., Tomita, M., Tsutsumi, J., Aoki-Waibel, N., Waibel, A., and Ward, W. Recent Advances in JANUS: A Speech Translation System. In *[EUROSPEECH, 1993]*, volume 2, pages 1295–1298.

[Woszczyna and Finke, 1996] Woszczyna, M. and Finke, M. Minimizing Search Errors Due to Delayed Bigrams in Real-Time Speech Recognition Systems. In *[ICASSP, 1996]*, volume 1, page 137.

[Woszczyna and Fritsch, 1997] Woszczyna, M. and Fritsch, J. Codebuchübergreifende Bucket-Box-Inersection zur schnellen Berechnung von Emissionswahrscheinlichkeiten im Karlsruher VM-Erkenner. Technical Report vm-report-211-97, Verbmobil-2.

[Z. Li and O'Shaughnessy, 1996] Z. Li, M. H. and O'Shaughnessy, D. New Developments in the INRS Continuous Speech Recognition System. In *[ICSLP, 1996]*, volume 1, pages 2–5.

Curriculum Vita

I was born on October 11th, 1966, in Brussels, Belgium. Since my parents are German, I have German citizenship. In 1968 our family moved to Luxembourg, where I attended the German section of the European School from 1972 to 1984.

In October 1984 I enrolled at the department of physics of the Technical University of Karlsruhe, where I graduated in 1990. The *Diplom* in Physics is roughly equivalent to a master's degree. The thesis work between January 1989 and June 1990 was on thermoelectric infrared sensors for deep space exploration.

During my time at the department of physics I tutored several student groups and labs and started to develop commercial software for a variety of companies.

Throughout my career, I have been committed to maintaining and extending the excellent language skills provided by the European School. I am fluent in German, English and French and have a working knowledge of Japanese.

Since 1991 I have been affiliated with the Interactive Systems Laboratories, working on speech recognition and translation for Prof. Dr. A. Waibel. Most of the time I have been working at the section at the department of computer science of Karlsruhe University, interrupted by regular stays at the section at the Carnegie Mellon University in Pittsburgh.

Publications

- **Codebuchübergreifende Bucket-Box-Inersection zur schnellen Berechnung von Emissionswahrscheinlichkeiten im Karlsruher VM-Erkenner**
  M. Woszczyna and J. Fritsch
  Verbmobil-Report 211-79, July 1997.

- **JANUS-II: Translation of Spontaneous Conversational Speech**
  A. Waibel, M. Finke, D. Gates, M. Gavald, T. Kemp, A. Lavie, L. Levin, M. Maier, L. Mayfield, A. McNair, I. Rogina, K. Shima, T. Sloboda, M. Woszczyna, T. Zeppenfeld and P. Zhan
  in the proceedings of the ICASSP 1996.

- **Minimizing Search Errors Due to Delayed Bigrams in Real-Time Speech Recognition Systems**
  M. Woszczyna and M. Finke
  in the proceedings of the ICASSP 1996.

- **End-to-End Evaluation in JANUS: A Speech-to-Speech Translation System**
  D. Gates, A. Lavie, L. Levin, A. Waibel, M. Gavalda, L. Mayfield, M. Woszczyna and P. Zhan
  in the proceedings of the ECAI 1996, Budapest.

- **JANUS: Towards Multilingual Spoken Language Translation**
  B. Suhm, P. Geutner, T. Kemp, A. Lavie, L. Mayfield, A. McNair, I. Rogina, T. Sloboda, W. Ward, M. Woszczyna and A. Waibel
  in the proceedings of the ARPA Spoken Language Technology Workshops 1995, Austin, Texas, January 1995.

- **Integrating Different Learning Approaches into a Multilingual Spoken Language Translation System**
  P. Geutner, B. Suhm, T. Kemp, A. Lavie, L. Mayfield, A. McNair, I. Rogina, T. Sloboda, W. Ward, M. Woszczcyna and A. Waibel
  in the proceedings of the IJCAI-Workshop on New Approaches to Learning for Natural Language Processing, Montreal, August 1995.

- **Inferring Linguistic Structure in Spoken Language**
  M. Woszczyna and A. Waibel
  in the proceedings of the ICSLP 1994.

- **JANUS94: Towards Spontaneous Speech Translation**
  M. Woszczyna, N. Aoki-Waibel, F. D. Bu, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C.P. Rose, T. Schultz, B. Suhm, M. Tomita and A. Waibel.
  in the proceedings of the ICASSP 1994.

- **Recent Advances in JANUS: A Speech Translation System**
  A. Waibel and M. Woszczyna
  in Speech Recognition and Coding (NATO ASI Series, Springer).

- **Recent Advances in JANUS: A Speech Translation System**
  M. Woszczyna, N. Coccaro, A. Eisele, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rose,T. Sloboda, M. Tomita, J. Tsutsumi, N. Aoki-Waibel, A. Waibel and W. Ward
  in the proceedings of the ARPA-HLT Workshop 1993.

- **Detection and Transcription of New Words**
  B. Suhm, M. Woszczyna, and A. Waibel.
  in the proceedings of the Eurospeech 1993.

- **Recent Advances in JANUS: A Speech Translation System**
  M. Woszczyna, N. Coccaro, A. Eisele, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. P. Rose, T. Sloboda, M. Tomita, J. Tsutsumi, N. Aoki-Waibel, A. Waibel, and W. Ward.
  in the proceedings of the Eurospeech 1993.

- **Testing Generality in JANUS: A Multi-Lingual Speech Translation System**
  L. Osterholtz, C. Augustine, A. McNair, I. Rogina, H. Saito, T. Sloboda, J. Tebelskis, A. Waibel, and M. Woszczyna
  in the proceedings of the ICASSP 1992.

Acknowledgements

From too much love of living,
From hope and fear set free,
We thank with brief thanksgiving,
Whatever gods may be,
That no life lives forever,
That dead men rise up never,
That even the weariest river winds somewhere safe to sea.
– Swinburne

First of all, I want to thank Alex Waibel. Not because I probably should, but because he is responsible for shifting my interest from semiconductors to speech recognition. Also, because I intensely enjoy to work at a place where once in a while the impossible is made possible.

Wayne Ward took the time to read and discuss the whole thesis and managed to keep a positive and constructive mood.

I owe all authors of JRTk, most of all Ivica Rogina, who's *Wall-Street-Journal* system served as baseline for most of my experiments. Jürgen Fritsch and Detlef Koll have also actively supported my efforts towards a faster recognizer.

All my colleagues, who covered my back to give me time to concentrate on my thesis actually deserve more than a few words of thanks. Thomas Schaaf for instance, saved me from having to bother with the much feared *VerbMobil* integration that otherwise might well have taken another couple of months of my time. Because he sacrificed a lot of private time for this, I also have to apologize to his wife and newborn twins Jan and Phillip.

The first to read and comment this thesis were Thomas Kemp and Tilo Sloboda. The most constructive comments came from Detlef Koll, who convinced me to rewrite major portions just when I thought I was almost done. The English translation was revised by Arthur McNair and Laura Mayfield.

Now, I did not forget Michael Finke, he just gets a paragraph on his own for which he will probably hate me. Through his roles as consultant, researcher, developer, programmer, and pace-setter he really makes things happen.

Last, but not least, I would like to point out that there are quite a few people who deserved my gratitude for reasons that do not quite belong into the framework of a scientific publication. I promise to make sure they all know I have not forgotten what they have done.

Index

A-Star Search, 30
Abbot
    Neural Nets, 48
    System Overview, 118
Abstract
    English, 1
    German, 3
Active Word Ends
    Experiments, 65
    Maximum Number of, 63
Allophones, 22, 23
Applications, 9, 101, 108
AT&T
    System Overview, 115

Backtrace
    for Linear Search, 59
    for Word Graphs, 44
    Introduction, 28
Bayes, 17
BBN
    Byblos, 116
    System Overview, 116
Beams
    Experiments, 63
    Funamentals, 39
    More than one, 61
    Problems, 39
Benchmarks
    Compiler Options, 51
    Experiments, 51
    Introduction, 51
Big-BBI, 87
Bigrams, 17, 36, 37
Bucket Box Intersection
    Experiments, 84
    Introduction, 82

Cambridge University
    Abbot, 48, 118
    HTK, 119
CMU
    SPHINX, 120
    System Overview, 120
Codebook, 23
    for Gaussian Selection, 46
    size for NAB Recognizer, 34
Compiler Options, 51
Conclusion, 111
Conditional frame skipping, 99
Confidence Interval
    Example, 96
    Recognition Speed, 16
    Word Accuracy, 15
Continuous Speech Recognition, 26
    Limitations, 10
Contributions, 55, 111
CPU-time, 16, 51
Cross-Word Models, 56
    Leaf Copies, 58
    Left Context, 56
    Multiplexed root nodes, 56
    Right Context, 58
    Single Phone Words, 58

Daimler Benz
    System Overview, 121
Data Reduction, 18, 45
Delayed Bigrams
    Experiments, 69
    Introduction, 37
Depth First Search, 30
Distribution Weights, 23
Dragon
    Naturally Speaking, 113, 121
    System Overview, 121
DTW, 27

Entropics, 119
Evaluation Data, 14, 33

Fast Match
    Experiments, 73
    Introduction, 71
Fastest System, 98
Feature Vector, 19
Flat Search, 59
Foreign Accents, 106
Forward Algorithm, 23

Forward-Backward Search, 43
Frame skipping, conditional, 99

Gaussian Selection, 46
Generalized-BBI
    Back-Off Strategies, 90
    Introduction, 87
    Parameters, 93
    Summary, 94
    with all Gaussians, 88
    with Nearest Neighbor, 90
German Spontaneous Scheduling Task, 102
GSST, 102

Hardware, 50, 51
Hidden Markov Models, 22
HTK
    System Overview, 119

IBM
    System Overview, 122
    ViaVoice, 113, 122
Introduction, 9

JANUS, 31
JANUS-3, 31
JRTk, 31

Language Models
    Back-offs, 36
    Delayed Bigrams, 37
    for NAB, 34
    Introduction, 17
    Smoothing, 36
    Speed Issues, 36
    Unigram Lookaheads, 69
LDA
    for Reduced Dimensionality, 78
LIMSI
    System Overview, 123
Linear Search, 59
    Experiments, 60
Low Level Optimizations, 81

Microphone Independence, 106
Microsoft
    System Overview, 124
    Whisper, 124
MIT
    Lincoln Stack-Decoder, 125
    System Overview, 125

N-Best Search, 29
NAB, 33
    P0 evaluation, 33
    Spoke Conditions, 33, 105
Nearest Neighbor Approximation
    Experiments, 46
    Introduction, 45
Neural Nets
    ABBOT, 118
North-American-Business-News, 33
NOWAY stack decoder, 118

Observation Probabilities
    BBI, 82
    Generalized-BBI, 87
    Hardware, 50
    in JRTk, 32
    Introduction, 23
    Neural Nets, 48
    Profiling, 76
    Speed Issues, 45
One Stage Dynamic Time Warping, 27
Order of the Gaussians, 77

Parallelizing
    for Special Hardware, 50
    with Threads, 49
Philips
    System Overview, 125
Phoneme, 21
Phoneme-Lookaheads
    Experiments, 73
    in JRTk, 72
    Introduction, 71
Pipelining, 107
Polyphones, 22
Portability
    Experiments, 101
    to GSST, 102
    to NAB male speakers, 102
Preprocessing
    and Pipelining, 107
    for NAB System, 34
    Introduction, 19
Pronunciation Dictionaries, 21
Pronunciation Variants, 21
Pruning, 29
    Active word ends in JRTk, 63
    Experiments, 63
    Introduction, 39
    with several Beams, 61
PTM, 117

Recognition, 26
Recognition Speed
    Definition, 16

Fastest System, 98
Original System, 34
Reduced Dimensionality
Experiments, 79
Introduction, 78
Robustness, 105
Foreign Accents, 106
Microphone Independence, 106
RT-Factor, 16
Run-on recognition, 107

Sampling, 18
Sampling-Rate, 18
SCTM, 117
Search
A-Star, 30
Depth First Search, 30
Forward-Backward Search, 43
Introduction, 26
Speed Issues, 39
Tree Copies, 42
Tree Search, 41
Unigram Lookaheads, 69
Siemens
System Overview, 126
Siemens Speech Recognition, 113
Significance, 15
Single phone words, 58
Skipping
Input Frames, 98
Score Computations, 96
SPECfp92, 51
Spectrum, 19
Speech Recognition, 13, 17
Speech Recognition Problem, 17
Speed Issues, 35
Spontaneous Speech, 15, 102
SRI
System Overview, 126
Sun-Ultra, 51
System description
GSST Recognizer, 102
NAB Recognizer, 34
System Overview, 115
Abbot, 118
AT&T, 115
BBN, 116
Daimler Benz, 121
Dragon, 121
HTK, 119
IBM, 122
LIMSI, 123
Microsoft, 124
MIT, 125
Other, 127
Philips, 125
Siemens, 126
SPHINX, 120
SRI, 126
University of Aachen, 125

Task Description, 33
Test Data, 14, 33
Test Environment, 31
Threads, 49
Tradeoffs, 10
Training Data, 14
Tree Copies, 42
Tree Search
Implementation, 58
Introduction, 41
Trigrams, 17, 59

Unigram-Lookaheads
Experiments, 69
Introduction, 69

VerbMobil, 102
Viterbi Algorithm, 25
Vocabulary Size, 33

Weights, 23
Word Accuracy, 15
Word Correct Rate, 15
Word Error Rate, 15
Word Graphs, 44
Workstations, 51